Vysoká škola báňská – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

INET Framework pro simulační nástroj OMNeT++

INET Framework for OMNeT++

2019 **Lukáš Bik**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Zadání bakalářské práce

Student: **Lukáš Bik**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2601R013 Telekomunikační technika

Téma: INET Framework pro simulační nástroj OMNeT++
INET Framework for OMNet++

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem práce je popis simulačního systému OMNeT++ a frameworku INET. Dalším výstupem práce je vytvoření 2 úloh využívající tento framework, úlohy budou využity ve cvičení odborného předmětu.
1. Popište možnosti INET Framework pro simulační nástroj OMNet++.
2. Navrhněte a realizujte 2 komplexní simulační příklady využívající tento framework.
3. Otestujte toto řešení a celý postup zdokumentujte formou zadání laboratorního cvičení do odborného předmětu.

Seznam doporučené odborné literatury:

[1] WEHRLE, Klaus; GÜNES, Mesut; GROSS, James. *Modeling and Tools for Network Simulation*. 1st Edition. Germany : Springer, 2010. 256 s. ISBN 978-3642123306.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Libor Michalek, Ph.D.**

Datum zadání: 01.09.2019
Datum odevzdání: 30.04.2020

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA Fakulta elektrotechniky a informatiky ID16 OSTRAVA

prof. Ing. Miroslav Vozňák, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlášení studenta

Prohlašuji, že jsem tuto bakalářskou/diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne: *13. května 2020*

……..……
podpis studenta

Poděkování

Rád bych poděkoval vedoucímu bakalářské práce Ing. Liborovi Michalkovi, Phd. za odbornou pomoc a vždy pozitivní přístup při řešení problémů týkajících se této práce.

Abstrakt

Cílem práce je seznámení studenta s možnostmi simulačního prostředí OMNeT++ a to zejména s jeho rozšířením INET Framework. V úvodu je čtenář teoreticky seznámen se základními prvky simulačního nástroje. V práci je taktéž uveden postup pro instalaci užívaného simulačního prostředí i s potřebným rozšířením. Značná část pak popisuje strukturu některých prvků, které následně student používá v navržených praktických simulačních příkladech. Tyto simulační příklady studentovi přiblíží práci se simulačním programem OMNeT++, funkci a implementaci užitých modulů, které jsou součástí rozšíření INET Framework a v poslední řadě představí studentovi chování některých užitých směrovacích protokolů.

Klíčová slova

OMNeT++, INET Framework, Eclipse, simulační prostředí, RIP, OSPF

Abstract

The aim of the thesis is to acquaint students with the possibilities of the OMNeT++ simulation environment, especially with its extension INET Framework. In the introduction, the reader is theoretically informed about the basic elements of the simulation tool. The work also presents the procedure of installation of the simulation environment with the required extension. A significant part of the work then describes the structure of some elements which students subsequently use in the designed practical simulation exercises. These simulation exercises show the work with the programme, function and implementation of modules included in the extension INET Framework and introduces the behaviour of some of the routing protocols.

Key words

OMNeT++, INET Framework, Eclipse, simulation environment, RIP, OSPF

Obsah

1 Úvod

Tato bakalářská práce má za úkol seznámit čtenáře se simulačním prostředím OMNeT++, a to zejména s jeho doplňkem INET Framework. Použití tohoto INET Framework umožní uživateli simulovat různé scénáře v oblasti sítí, síťových topologií a jejich síťových komponentů. Simulace jako taková zejména v dnešní době nabývá důležitosti. Umožní uživateli testovat navrženou topologii sítě ještě před její realizací. Což je v dnešní době, tedy v době rozsáhlých podnikových sítí, nemálo důležitá vlastnost. Rovněž co se bezpečnosti týče mohou být simulační nástroje dobrým sluhou. Uživatel má k dispozici širokou škálu možností práce se simulačním prostředím, ať už testování chodu a spolehlivosti jednotlivých síťových zařízení a protokolů až po simulaci DoS útoků.

Simulace, které nám OMNeT++ a další jemu podobné programy umožní mají nezměrný výukový potenciál. Studenti zde mohou podrobně pozorovat funkci síťových zařízení a celkově pochopit myšlenku přenosu dat po síti. Tato simulační prostředí umožní nejen pozorovat funkci směrovacích protokolů, ale taktéž dovolí studentovi bez rizika konfigurovat různá síťová zařízení. Student si tedy může síť navrhnout, v programovém prostředí realizovat a následně i nakonfigurovat a odzkoušet.

OMNeT++ i veškeré jeho komponenty jsou naprogramovány v jazyce C++. Je využíván zejména k síťové simulaci, ovšem nejedná se o jediné možné využití. OMNeT++ může uživateli taktéž nabídnout simulaci mobilních sítí (LTE simulaci), optických datových sítí, ale také simulaci práce s HPC (High-performance computer) systémy. Nicméně tato bakalářská práce bude zaměřena pouze na využití rozšíření INET Framework k simulaci datových sítí. Výhodou programu OMNeT++ je jeho grafické rozhraní které lze přidružit k prostředí Eclipse. Toto prostředí nám umožní vizuálně pracovat s navrženou topologií, zobrazovat různé animace a debugovat, celkově uživateli značně usnadňuje práci se simulačním modelem.

Prostředí je spustitelné na operačním systému Windows, Linux nebo macOS. Na prvních dvou zmíněných tato bakalářská práce taktéž popíše instalaci. OMNeT je distribuován pod ACADEMIC PUBLIC LICENSE, která je podmínkami téměř totožná s dobře známou GNU General Public License. Jedná se tedy o „svobodný" software.

INET Framework je „open-source" model pro OMNeT++, který obsahuje moduly s implementací veškerých pro nás důležitých protokolů (IP, TCP, UDP, PPP, Ethernet a další). INET Framework je také využíván jako stavební kámen pro další přidružené frameworky. INET je založen na konceptu vzájemné komunikace mezi jeho moduly.

V poslední řadě se tato bakalářská práce bude zaobírat otázkou praktického rázu. INET Framework využijeme k návrhu simulačních příkladů, jejichž podobu následně zpracujeme do formy využitelné jako zadání laboratorního cvičení v odborném předmětu jež se věnuje tématice počítačových sítí.

2 Základní vlastnosti OMNeT++

OMNeT++ je rozšířitelný, objektově založený a modulární simulační nástroj pracující na principu diskrétní simulace. Pro usnadnění práce s jeho moduly nabízí OMNeT++ uživateli grafické rozhraní. Velká výhoda tohoto nástroje spočívá v jeho rozšiřitelnosti několika modely. Tedy místo aby developeři vytvořili jeden úzce specializovaný simulátor, navrhli OMNeT++ tak, aby byl schopen zaštítit různé modely třetích stran. Navzdory tomu je OMNeT++ často mylně označován jako simulátor počítačových sítí. Tato možnost využití přichází až s jedním nejznámějším a nejvíce využívaným modelem INET Framework [1].

2.1 Princip diskrétní simulace

Základní myšlenka diskrétní simulace je skokový posun z jedné simulační události ke druhé. Každý takový „event" může mít za následek změnu hodnot, ze kterých je složen modul. Každá událost je zaznamenána v podobě oznámení o události (event notice) do listu události (future event list), který řídí chod všech událostí v simulaci. V oznámení o události figurují zejména dva důležité údaje. Čas (time) a Typ (type). Datový údaj Čas obsahuje konkrétní čas, kdy událost v naší simulaci nastane. Typ události je obsažen v proměnné Typ. List události obsahuje základní funkce na vložení, hledání či mazání jednotlivých záznamů v listu [1].

2.2 Modely a moduly

OMNeT++ a jeho funkcionalita je založena na přídavných modelech. Veškerá simulace tedy probíhá pod těmito modely. Modely se různí jejich funkcionalitou, která nemusí být nutně síťového charakteru. Většina modelů jsou jako open source projekty vyvíjeny nezávisle rozsáhlou komunitou samotného OMNeT++.

Všechny tyto modely jsou skládány z jednotlivých modulů. Modulem lze považovat například jednotlivé směrovací protokoly, user agenty, směrovací tabulky, ale i jednotlivé simulační funkce, například automatické přidělování IP adres v síti. Samotné prvky sítě jako koncový uživatel, směrovače, přepínače jsou výsledkem skládání jednoduchých modulů do skupin (compound modules). Moduly mezi sebou komunikují vzájemným zasíláním zpráv. Implementace modulů je řešena jazykem C++., využitím simulačních knihoven [1].

2.2.1 Jednoduchý modul

Jednoduché moduly jsou aktivními komponenty každého modelu. Jejich implementace je v jazyce C++ a využívá knihoven z třídy OMNeT++. Jednoduché moduly nelze dále dělit, lze je považovat za základní. Chování jednoduchých modulů je definováno uživatelem a implementací v jazyce C++. Jednoduchý modul není nic jiného než C++ třída, která musí být podtřídou třídy cSimpleModule s minimálně jednou virtuální metodou, která definuje chování našeho jednoduchého modulu. Jednoduché moduly generují, nebo reagují na události simulace. V síťových simulacích mohou jednoduché moduly reprezentovat síťová zařízení, síťové protokoly jako TCP, UDP, nebo datové struktury v podobě směrovacích tabulek [2].

```
// file: HelloModule.cc
#include <omnetpp.h>
using namespace omnetpp;

class HelloModule : public cSimpleModule
{
  protected:
    virtual void initialize();
    virtual void handleMessage(cMessage *msg);
};
Define_Module(HelloModule); // registruje modul do OMNeT++
void HelloModule::initialize()
{
    EV << "Hello World!\n";
}
void HelloModule::handleMessage(cMessage *msg)
{
    delete msg;
}
```

Příklad implementace jednoduchého modulu HelloModule [2]

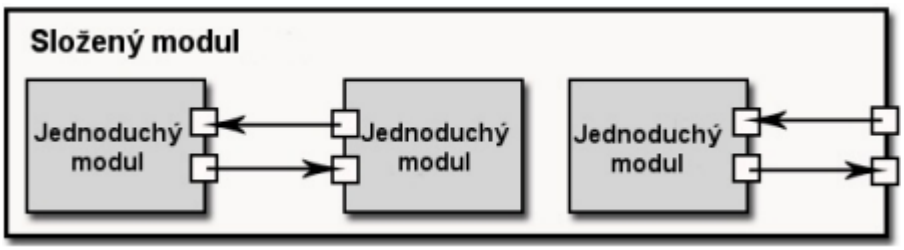
K umožnění přístupu k této implementaci jednoduchého modulu je třeba zajistit konektivitu s NED soubory. Možná NED deklarace může být provedena následujícím způsobem.

```
// file: HelloModule.ned
simple HelloModule
{
    gates:
        input in;
}
```

Asociativní NED deklarace našeho jednoduchého modulu HelloModule [2]

2.2.2 Složený modul

Jediný účel složeného modulu je logické uspořádání simulačního systému. Je složen z několika vzájemně komunikujících modulů. Tyto složené moduly negenerují žádné události, proto je považujeme za neaktivní. Složením jednoduchých modulů můžeme například simulovat chování směrovačů a dalších síťových bodů. Složené moduly mohou mít taktéž jako jednoduché moduly parametry a brány sloužící k vzájemnému posílání zpráv.



Obrázek 2.1: Princip spojování modulů [3]

2.2.3 Zasílání zpráv

Vzájemná komunikace mezi moduly je řešena pomocí zasílání zpráv. Tyto zprávy reprezentují fyzický signál na přenosovém médiu. Jednotlivé moduly mohou zprávy posílat buďto rovnou cílovému modulu, nebo mohou cestu definovat pomocí bran. Brány se starají o vzájemnou konektivitu mezi moduly a lze je považovat za vstupní a výstupní rozhraní modulů [4]. Zprávy jsou reprezentovány třídou cMessage a mohou reprezentovat například paket nebo rámec [5].

2.3 NED Jazyk

OMNeT++ má svůj doménově specifický jazyk s názvem NED, který slouží k popisu struktury celého modelu, nebo jednotlivých modulů. Prvotní nápad použít k tomuto popisu jeden z již dobře fungujících jazyků jako XML, Python nebo Ruby byl nakonec odmítnut. Modely popsány tímto způsobem by nebyly kompatibilní s grafickým editorem. NED tedy popisuje deklaraci jednoduchých modulů, definici složitých modulů a definici struktury sítě. Deklarace jednoduchých modulů popisuje interface modulu, tedy brány a jeho parametry. Popis složených

modulů obsahuje deklaraci jednotlivých jednoduchých modulů a popis jejich vzájemných propojení. Pro udržení přehledu i v případě rozsáhlých simulačních scénářích používá NED jazyk Java-style package systém [6].

3 INET Framework

INET Framework je open-source modelová knihovna pro OMNeT++. Poskytuje uživateli moduly, jejichž implementace simuluje funkci reálných síťových prvků. Koncept modulů je založen na jejich vzájemné komunikaci zasíláním zpráv. INET je znám širokou škálou možného využití v oblastí komunikačních sítí, ať už bezdrátových, mobilních, ale i sítí senzorů. Obsahuje moduly s funkcí směrovacích protokolů (OSPF, BGP, TCP, UDP, IPv4, IPv6, atd.) nebo i protokolů linkové vrstvy (Ethernet, PPP, IEEE 802.11, atd.). INET Framework využívají i některé další simulační frameworky jako základ, který pak rozšiřují vlastním směrem [7].

3.1 Simulace

Základ simulace tvoří modul reprezentující počítačovou síť. Síť je tvořena složenými moduly, které obsahují síťové prvky a automatické síťové konfigurátory. Síť je dále tvořena spojeními mezi jednotlivými síťovými prvky. Tato spojení reprezentují kabelové spoje. Rozsáhlé hierarchické sítě mohou být také zakomponovány do složených modulů. Moduly jsou uspořádány v adresáři. Struktura tohoto uspořádání přibližně odpovídá vrstvám referenčnímu ISO/OSI modelu [8].

src/inet/applications/ – generátory síťového provozu a aplikační modely
src/inet/transportlayer/ – protokoly transportní vrstvy
src/inet/networklayer/ – protokoly síťové vrstvy
src/inet/linklayer/ – protokoly linkové vrstvy
src/inet/physicallayer/ – protokoly fyzické vrstvy
src/inet/routing/ – směrovací protokoly (internetové i ad hoc)
src/inet/mobility/ – modely mobility
src/inet/power/ – nástroje na modelování spotřeby energie
src/inet/environment/ – model fyzického prostředí
src/inet/node/ – předem sestavené modely síťových uzlů
src/inet/visualizer/ – vizualizační komponenty (2D a 3D)
src/inet/common/ – různé další užitečné nástroje

Uživatel při modelování síťové topologie využívá převážně NED soubory a INI soubory ke konfiguraci. INET uživateli nabízí předpřipravené síťové uzly s vybranými komponentami. Tyto moduly lze dále upravovat pomocí parametrů [9].

StandardHost – Modul reprezentující koncové zařízení v síťové topologii. Obsahuje nejběžnější internetové protokoly (UDP, TCP, IPv4, IPv6, Ethernet, IEEE 802.11). V modulu je taktéž zahrnut mobility model, optional energy model a různé aplikace, které jsou konfigurovatelné pomocí INI souborů.

EtherSwitch – Modeluje funkci ethernetového switche.

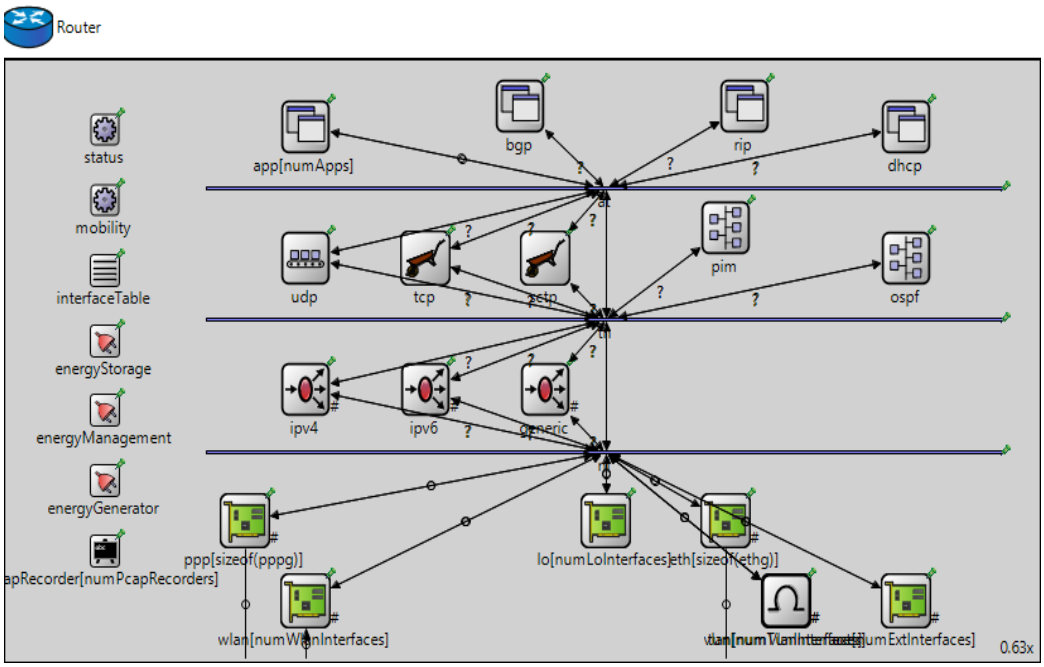
Router – Poskytuje uživateli nejběžnější směrovací protokoly (OSPF, RIP, BGP, PIM).

AccessPoint – Modeluje přístupový bod Wifi s několika IEEE 802.11 rozhraními a několika ethernetovými porty.

WirelessHost – Poskytuje síťový bod s klasicky jedním IEEE 802.11 síťovým rozhraním. Vhodný pro kombinaci s Acesspoint modulem.

3.2 Struktura složeného modulu router

Jak již bylo zmíněno, INET uživateli nabízí řadu síťových uzlů v podobě modulů. Model pro router (směrovač) v simulaci podporuje bezdrátové spojení, ethernet i PPP. Modul lze propojit pomocí ethernetového rozhraní s ostatními uzly sítě přes ethg bránu. Ve výchozím nastavení jsou umožněný pouze full-duplex spojení. Výchozí nastavení modulu taktéž nepodporuje bezdrátové přenosy. Při dynamickým směrování je třeba specifický směrovací protokol přidat nastavením jednoho z parametrů hasOspf/hasRip/hasBgp [10].



Obrázek 3.1: Struktura směrovače v grafickém editoru

Jedná se o složený modul, jehož prvky jsou moduly spadající do linkové, síťové, transportní a aplikační vrstvy. Přidáním modulů reprezentujících síťové protokoly (bgp, rip, dhcp, pim, ospf) vzniká složený modul reprezentující směrovač.

3.3 Síťová rozhraní

V simulaci prostřednictvím INET Framework jsou primárně pojmem síťová rozhraní nazývané prostředky komunikace mezi dvěma síťovými uzly. Rozhraní (interface) tedy reprezentují požadovanou kombinaci softwarových i hardwarových prvků požadovaných ke komunikaci. Různá síťová rozhraní jsou implementována jako složené moduly odpovídající modulu „INetworkInterface". INetworkInterface obsahuje několik podmodulů, pro nás

nejzajímavější jsou moduly „ IWiredInterface" a „ IWirelessInterface". Veškeré síťové rozhraní lze tedy rozdělit do dvou kategorií „bezdrátová síťová rozhraní" a „drátová síťová rozhraní" [11].

3.3.1 Vestavěná síťová rozhraní

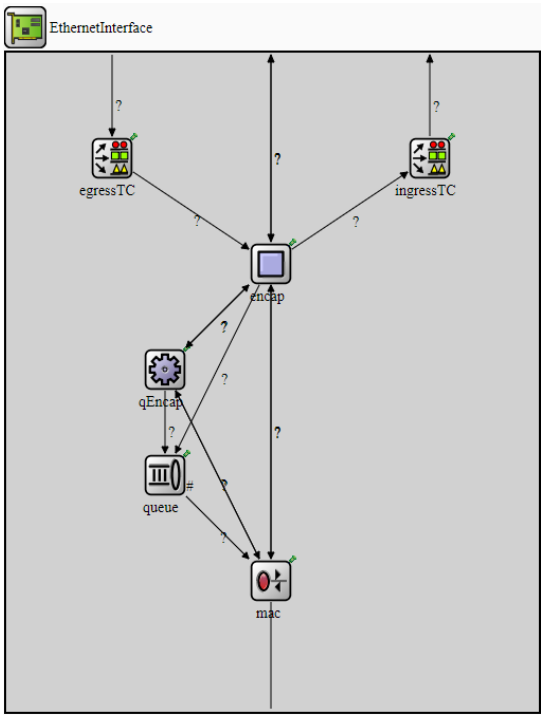
INET uživateli poskytuje předem připravená síťová rozhraní pro několik klasických protokolů [12].

Výčet nejběžnějších síťových rozhraní:

EthernetInterface – reprezentuje ethernetové rozhraní
PppInterface – rozhraní pro kabelové spoje užívající PPP komunikační protokol
Ieee80211Interface – reprezentuje rozhraní Wifi (IEEE 802.11)
Ieee802154NarrowbandInterface, Ieee802154UwbIrInterface – reprezentuje IEEE 802.15.4 rozhraní
LoopbackInterface – poskytuje loopback síťových uzlů

3.3.1.1 Struktura EthernetInterface rozhraní

EthernetInterface je složený modul zahrnující moduly EtherMac a EtherEncap. EtherEncap je implementován separátně z důvodu, že ne vždy je třeba provádět encapsulaci/decapsulaci. Obsahuje konfigurovatelné inpupt/output filtry typu IHook podobně jako modul PppInterface. INET Framework dále obsahuje dva MAC moduly pro ethernet. Modul „EtherMacFullDuplex" podporuje pouze full-duplex spojení. „EtherMac" implementuje úplnou MAC funkcionalitu (CSMA/CD, half-duplex, full-duplex, atd.) [13].



Obrázek 3.2: Struktura EthernetInterface

EthernetInterface obsahuje:

egressTC, ingressTC – Moduly nazývané „traffic conditioner". Klasifikují příchozí pakety a monitorují přenos mezi třídami.

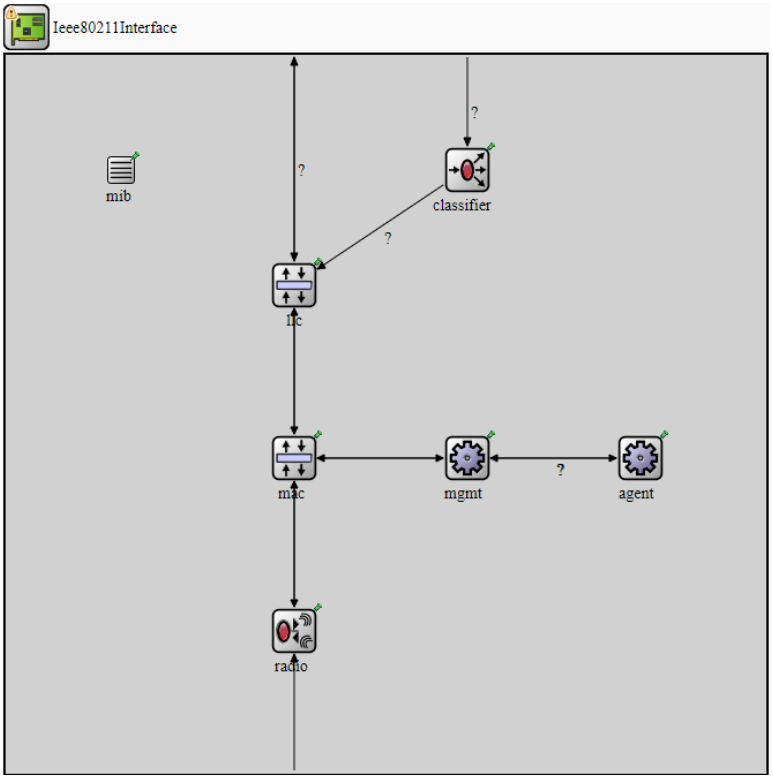
encap, qEncap – Moduly zajišťující encapsulaci/decapsulaci.

mac – Rozhraní pro MAC implementaci. Nezajišťuje encapsulaci ani decapsulaci. Zpracovává rámce přijaté z vyšší vrstvy, nebo rovnou ze sítě. Aplikuje CSMA/CD protokol. Provádí CRC kontrolu.

3.3.1.2 Struktura Ieee80211Interface rozhraní

Složený modul obsahuje implementaci IEEE 802.11. Účel tohoto standardu pro Wifi je modulace signálu do podoby vhodné pro rádiový přenos. Pro přenos jsou běžně používány pásma 2,4 GHz (802.11b a 802.11g) a 5 GHz (802.11a).

V INET přidáním Ieee80211Interface síťovém uzlu vzniká Wifi. INET má tyto prvky opět předpřipravené pod uzly WirelessHost, nebo AccessPoint. Toto rozhraní lze konfigurovat skrze mgmt.typename parametr buď do módu prostého přístupového bodu (access point), STA antény nebo módu ad-hoc sítě. Tento klasifikátor je třeba užít v momentě potřeby užití QoS [14][15].



Obrázek 3.3: struktura IEEE 802.11 rozhraní

Ieee80211Interface obsahuje:

classifier – Rozhraní pro QoS klasifikátor. Pro každý paket počítá uživatelskou prioritu (User Priority). Tuto hodnotu pak paketu před odesláním přiřadí.

llc – Rozhraní pro LLC modul.

mac – Rozhraní pro mac modul, který zajišťuje přenos rámců v rámci IEEE 802.11 standardu.

mgmt – Rozhraní pro všechny IEEE 802.11 moduly typu management. Specifikuje, které brány budou moduly managementu používat.

agent – Rozhraní agenta, iniciujícího vyhledávání kanálu nebo připojení a odpojení k přístupovým bodům sítě.

radio – Rozhraní pro moduly rádiového přenosu.

3.4 Směrovací protokoly

INET Framework obsahuje několik modulů pro internetové směrovací protokoly (RIP, OSPF, BGP), které definují způsob výměny informací mezi směrovači. Přidaný směrovač do simulované topologie má všechny instance těchto protokolů ve výchozím stavu nastaveny na false. INET taktéž obsahuje NED implementaci s už aktivním směrovacím protokolem ve výchozím stavu směrovače (RipRouter, OspfRouter, BgpRouter) [16].

3.4.1 RIP

RIP (Routing Information Protocol) je směrovací protokol typu distance-vector (vektor vzdálenosti), který používá počet skoků (hop count) jako metriku směrování. Proti smyčkám při směrování RIP preventivně implementuje limit na počet skoků (směrovačů) od zdroje k cíli.

RIP modul implementuje distance-vector směrování podle standardu RFC 2453 (RIPv2) nebo RFC 2080 (RIPng). Funkci lze vybrat nastavením parametru „RIPv2" nebo „RIPng". Konfigurace protokolu může být dále specifikována v XML užitím „ripConfig" parametru [17].

```
**.hasRip = true
**.mode = "RIPv2"
**.ripConfig = xmldoc("RIPConfig.xml")
```

Příklad konfigurace RIPv2 modulu směrovače [17]

3.4.2 OSPF

OSPF (Open Shortest Path First) je směrovací protokol typu link-state, každý směrovač zná strukturu celé sítě. Spadá do skupiny IGP (interior gateway protocols) fungující v rámci autonomního systému.

OSPF modul implementuje OSPF protokol verze 2. Oblasti a jednotlivé směrovače lze nakonfigurovat užitím XML souboru, který je opět třeba specifikovat jako parametr v „ospfConfig". Opět je možné užít buď modul klasického směrovače a následně nastavit parametr „hasOspf" na hodnotu true, nebo rovnou užít modul „OspfRouter" s již připraveným OSPF protokolem [18].

```
<?xml version="1.0"?>
<OSPFASConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:schemaLocation="OSPF.xsd">
  <!-- Areas -->
  <Area id="0.0.0.0">
    <AddressRange address="H1" mask="H1" status="Advertise" />
    <AddressRange address="H2" mask="H2" status="Advertise" />
    <AddressRange address="R1>R2" mask="R1>R2"
status="Advertise" />
    <AddressRange address="R2>R1" mask="R2>R1"
status="Advertise" />
  </Area>
  <!-- Routers -->
  <Router name="R1" RFC1583Compatible="true">
    <BroadcastInterface ifName="eth0" areaID="0.0.0.0"
interfaceOutputCost="1" routerPriority="1" />
    <PointToPointInterface ifName="eth1" areaID="0.0.0.0"
interfaceOutputCost="2" />
  </Router>
  <Router name="R2" RFC1583Compatible="true">
    <PointToPointInterface ifName="eth0" areaID="0.0.0.0"
interfaceOutputCost="2" />
    <BroadcastInterface ifName="eth1" areaID="0.0.0.0"
interfaceOutputCost="1" routerPriority="2" />
  </Router>
</OSPFASConfig>
```

Příklad implementace ospf v .xml souboru [18]

3.4.3 **BGP**

BGP je dynamický směrovací protokol užívaný opět v rámci autonomních systémů. Směrovače s nastaveným BGP směrováním dokážou automaticky reagovat na změny topologie sítě. INET poskytuje „BgpRouter", tedy modul směrovače s již předem nastaveným BGP směrováním. Modul má implementované BGP směrování verze 4 se standardem RFC 4271. Autonomní systémy a další parametry a pravidla lze opět konfigurovat v externím XML souboru [19].

4 Instalace OMNeT++ a INET Framework

4.1 Úvod

OMNeT++ je podporován na operačních systémech:

Windows 7 a 10
MacOS
Linux Ubuntu 16.04 LTS, Fedora Core 25, Red Hat Enterprise, OpenSUSE 42

Simulace může být spuštěna na kterékoliv unixové distribuci obsahující C++ compiler. Nicméně IDE je omezeno pouze na zmíněné operační systémy [20].

4.2 Windows

OMNeT++ je k dispozici ke stažení na oficiálních OMNeT webových stránkách https://omnetpp.org/download/. Ujistěte se, že před stažením souboru zvolíte správnou verzi operačního systému na spodní liště. Stažený zip soubor obsahuje veškeré nutné soubory ke spuštění. Obsah staženého zipu zkopírujte do složky, kde chcete OMNeT++ instalovat. Je doporučeno vybírat složku bez bílých znaků. Proto Program Files není vhodný prostor pro instalaci OMNeT++ [20].

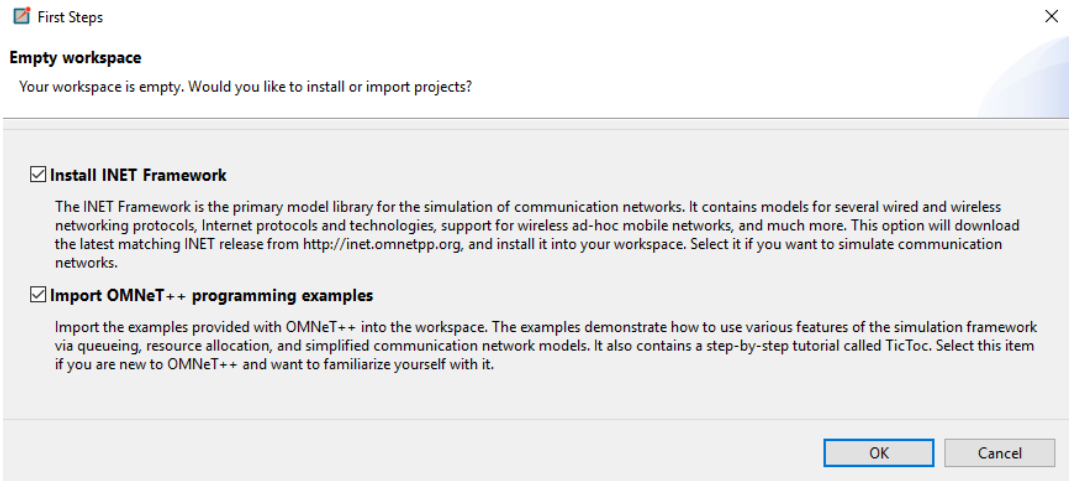
Přes příkazový řádek otevřete nově vytvořenou složku omnetpp-<verze>. Instalační nastavení lze měnit v konfiguračním souboru po zadání příkazu $ notepad configure.user, pro běžné potřeby poslouží výchozí nastavení.

Soubor mingwenv.cmd spusťte prostým příkazem $ mingwenv.cmd, pak potvrďte libovolnou klávesou. Následně se spustí extrahování nutných souborů. Po skončení opět potvrďte libovolnou klávesou. Otevře se nám sekundární konzole Mingw. Zde je třeba spustit build simulačních knihoven, proto píšeme následující příkazy:

```
$ ./configure
$ make
```

Následné příkazem $ omnetpp spustíte IDE.

Při následném spuštění IDE se zobrazí nabídka s možnou instalací pro vás potřebného INET Framework.



Obrázek 4.1: Nabízená instalace INET Framework

4.3 Linux Ubuntu

Jak již bylo zmíněno, instalace OMNeT++ je doporučována spíše na distribucích s c++ compilerem. Na ostatních, zmíněných distribucích instalace probíhá obdobně, tato kapitola ale bude zaměřena na instalaci na distribuci Ubuntu 16.04 nebo 18.04. Instalace bude popsána jednoduchými kroky v Linuxovém terminálu [20].

Než začnete s instalací, je doporučeno aktualizovat databázi balíčků příkazy:

```
$ sudo apt-get update
```

```
$ sudo apt-get upgrade
```

Pro instalaci potřebných balíčků zkopírujte do terminálu:

```
$ sudo apt-get install build-essential gcc g++ bison flex perl \ python python3 qt5-default   libqt5opengl5-dev tcl-dev tk-dev \ libxml2-dev zlib1g-dev default-jre doxygen graphviz libwebkitgtk-1.0
```

Pro umožnění paralelních simulací je taktéž vhodná instalace MPI balíčků:

```
$ sudo apt-get install openmpi-bin libopenmpi-dev (Do you want to continue [Y/N], odpověd Y)
```

OMNeT++ lze zadarmo stáhnout z oficiálních stránek https://omnetpp.org/download/, na šedé liště je opět nutné určit distribuci OS, na které pracujeme. Tentokrát zvolte Linux. Kliknutím na pole Download se stáhne omnetpp-5.5-src.tgz soubor. Obsah tohoto souboru je opět třeba rozbalit v místě, kde jej chcete nainstalovat. Obvyklé umístění je /home/<vy>/. Otevřete terminal a příkazem soubor rozbalte do aktuálního adresáře:

```
$ tar xvfz omnetpp-5.5-src.tgz
```

OMNeT++ potřebuje svůj bin/ adresář v linuxovém „path“. Toto potřebné nastavení provedete příkazy:

```
$ cd omnetpp-5.5
$ . setenv
```

Následně je třeba upravit konfigurační soubor .bashrc, lze provést tímto způsobem:

```
$ gedit ~/.bashrc
```

Na konec souboru následně umístěte řádek:

```
export PATH=$HOME/omnetpp-5.5/bin:$PATH
```

Nyní vypněte a zapněte terminal pro potvrzení změn. V nově otevřeném okně terminalu přejděte do kořenové složky OMNeT++. Zde napíšeme příkaz:

```
$ ./configure
```

Po uplynutí konfigurace už stačí pouze kompilovat, to proveďte takto:

```
$ make
```

Spustění OMNeT++ Simulation IDE lze provést příkazem:

```
$ omnetpp
```

Na závěr ještě doporučuji k usnadnění následujících přístupů k programu vytvořit na ploše application launcher příkazem:

```
$ make install-desktop-icon
```

5 OMNeT++ IDE

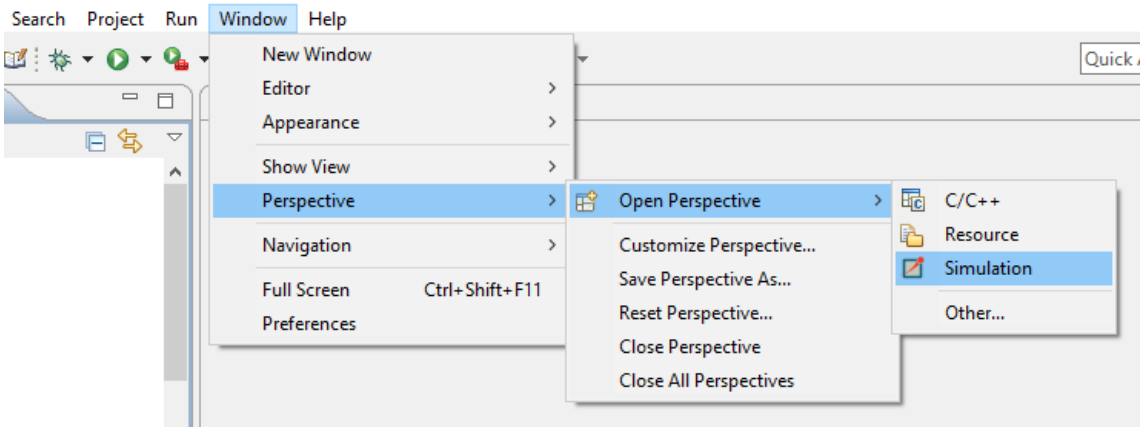
5.1 Úvod

Jak již bylo zmíněno, simulační prostředí programu OMNeT++ je postaveno na Eclipsovém základu. OMNeT++ navíc umožňuje vytváření nebo konfiguraci modelů (NED a INI soubory) nebo analýzu výsledků simulace [21].

5.2 Pracovní plocha

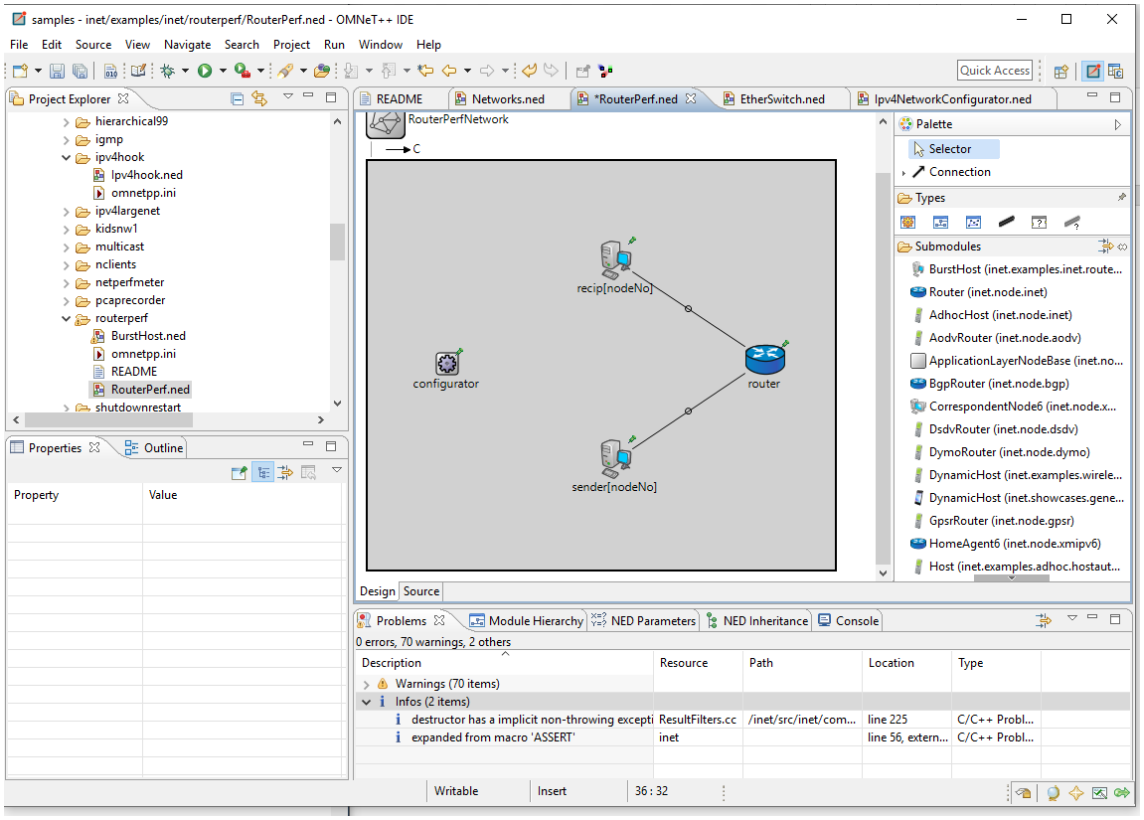
Úvodní obrazovka spuštěného IDE se skládá z editorů, nebo různých možností zobrazení. Ty jsou dále členěny do „ perspectives". Tyto perspektivy nám definují, která zobrazení a editory jsou na pracovní ploše viditelné.

OMNeT++ IDE poskytuje uživateli „Simulation perspective", které uživateli nabídne k práci příslušné simulační soubory [21].



Obrázek 5.1: Výběr simulační perspektivy

Samozřejmě volba perspektivy není nijak svazující. Pracovní rozhraní Eclipsu je variabilní. Jednotlivé viditelné elementy lze přeřadit, odebrat nebo nahradit. Je tedy na uživateli, jak bude jeho pracovní plocha v OMNeT++ IDE vypadat.



Obrázek 5.2: Náhled na OMNeT++ IDE

Průzkumník projektů v levém horním rohu zobrazuje uživateli projekty a jejich obsah. V příkladu uvedeném na obrázku 5.2 je otevřen soubor typu NED. Prostřední okno prostředí se ihned přizpůsobí typu spuštěného souboru. Lze tedy zobrazovat jak zdrojové kódy jednotlivých modulů, textové soubory nebo grafický náhled simulovaného scénáře. Na pravé straně prostředí je pak uživateli k dispozici paleta s užitečnými nástroji. Horní část palety je tvořena základními nástroji jako je ukazatel, ukazatel konektivity nebo nástroj pro tvorbu spojení. Prostřední část palety obsahuje základní elementy, které lze umístit na nejvyšší úroveň našeho NED souboru (jednoduchý modul, složený modul, kanál, atd.). Spodní část palety je tvořena všemi typy modulů, které lze užít jako submoduly. V levém dolním rohu jsou uživateli zobrazovaný vlastnosti objektu, který je vybrán z oblasti editoru [21].

5.3 NED editor

NED soubor vyhledáme v průzkumníku projektů a otevřeme jej. Jeho implementace se zobrazí v editoru v grafické podobě. Mezi módy úprav lze jednoduše přepínat pomocí „Design" a „Source" tlačítek na spodní liště pod editorem. Vybereme-li v grafickém zobrazení konkrétní prvek a přejdeme do zdrojového zobrazení, podtrhne se nám úvodní řádek jeho implementace [21].

5.3.1 Editace v grafickém módu

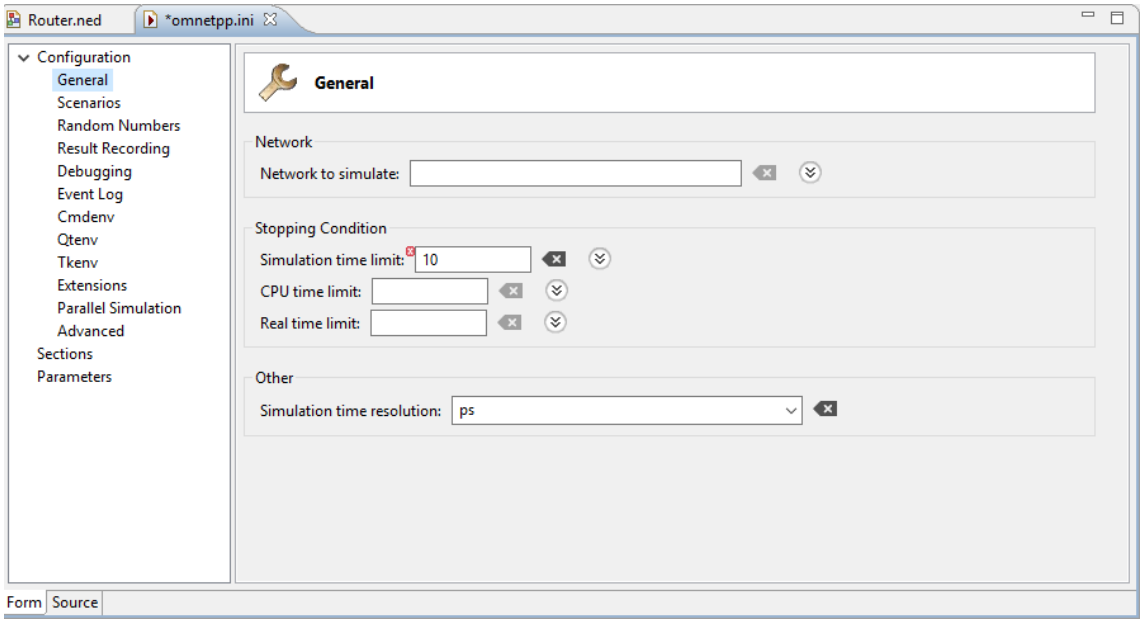
Grafický editor uživateli zobrazí viditelné elementy načteného NED souboru. Jednoduché moduly, složené moduly a sítě jsou v editoru reprezentovány ikonami. Každý NED soubor může obsahovat i více než jeden modul nebo síť. Propojení mezi jednotlivými moduly jsou znázorněna šipkami nebo linkami. Práce v grafickém módu uživateli umožňuje vytvářet a ovlivňovat simulované scénáře bez znalosti programování [21].

5.3.2 Editace v módu zdrojového kódu

Editor zdrojového kódu podporuje veškerou funkcionalitu očekávanou od editoru postaveném na Eclipsovém základu (vyhledávání, našeptávání atd.). Editor zdrojového kódu taktéž aktivně reaguje na nesrovnalosti a chyby ve zdrojovém kódu podtržením chybného řádku [21].

5.4 INI editor

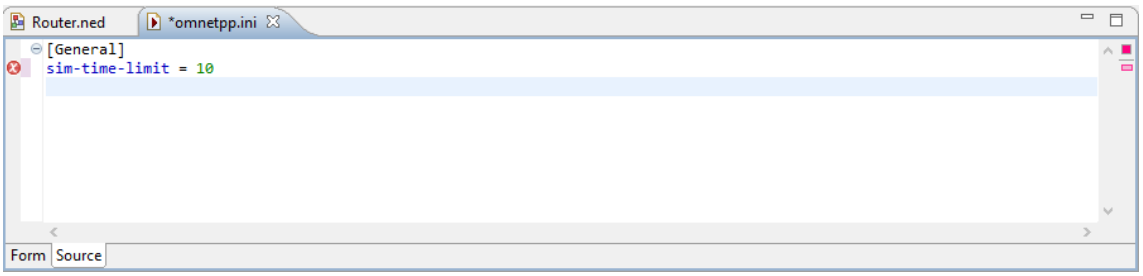
Simulační modely jsou parametrizovány a konfigurovány pomocí .ini souborů. .Ini soubory jsou textové soubory editovatelné v jakémkoliv textovém editoru. Přesto OMNeT++ obsahuje speciálně navržený editor na jejich úpravu. Ten umožňuje uživateli pohodlnou, a hlavně přehlednou úpravu .ini souborů. Editor má detailní znalost simulovaného modelu a všech dostupných konfiguračních nastaveních. Konfiguraci .ini souborů lze provádět jednoduchými formuláři a dialogy, nebo přímou úpravou textového souboru s konfigurací [21].



Obrázek 5.3: Editace .ini souborů pomocí dialogů

Opět v případě úpravy .ini souborů pomocí dialogů a formulářů není po uživateli požadovaná znalost programování. Lze postupovat intuitivně. Veškeré změny provedené

v grafické editaci se paralelně vypisují i v textovém editoru. Tento mód lze změnit opět přepnutím pomocí „Source" na spodní liště [21].



Obrázek 5.4: Editace .ini souborů pomocí textového souboru

6 Praktické simulační příklady využívající INET Framework

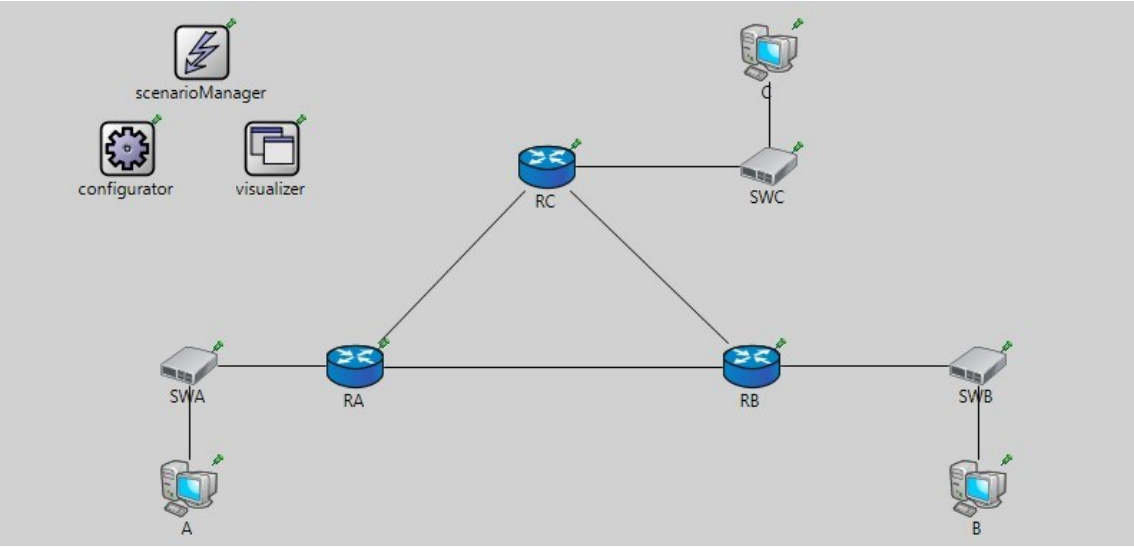
Tato kapitola má za úkol postupně představit práci se simulačním prostředím programu OMNeT++, a to konkrétně při práci s INET Framework. Čtenáři bude poskytnut přesný návod vedoucí k úspěšné realizaci požadovaných zadání.

6.1 Implementace směrovacího protokolu RIP

Úloha seznamuje čtenáře se základními funkcemi simulačního prostředí OMNeT++ a využívá při simulaci prvky INET Framework. Při realizaci zadání není požadovaná širší znalost směrovacího protokolu RIP.

6.1.1 Zadání

V simulačním prostředí OMNeT++ navrhněte topologii podle obrázku 6.1. Při návrhu využijte předpřipravené prvky RipRouter, EtherSwitch, StandardHost, Ipv4NetworkConfigurator, IntegratedCanvasVisualizer a scenarioManager ve sloupci „Submodules". Dodržte zadaný způsob pojmenovaní všech prvků topologie. Nakonfigurujte zařízení topologie pomocí modulu Ipv4NetworkConfigurator. Správnost sestavené topologie otestujte využitím PingApp pingem mezi zařízeními A, B. Ve třicáté vteřině běhu simulace užitím modulu scenarioManager odpojte linku mezi směrovači RA a RB a sledujte následnou reakci topologie na tuto událost.



Obrázek 6.1: schéma zapojení topologie s RIP protokolem

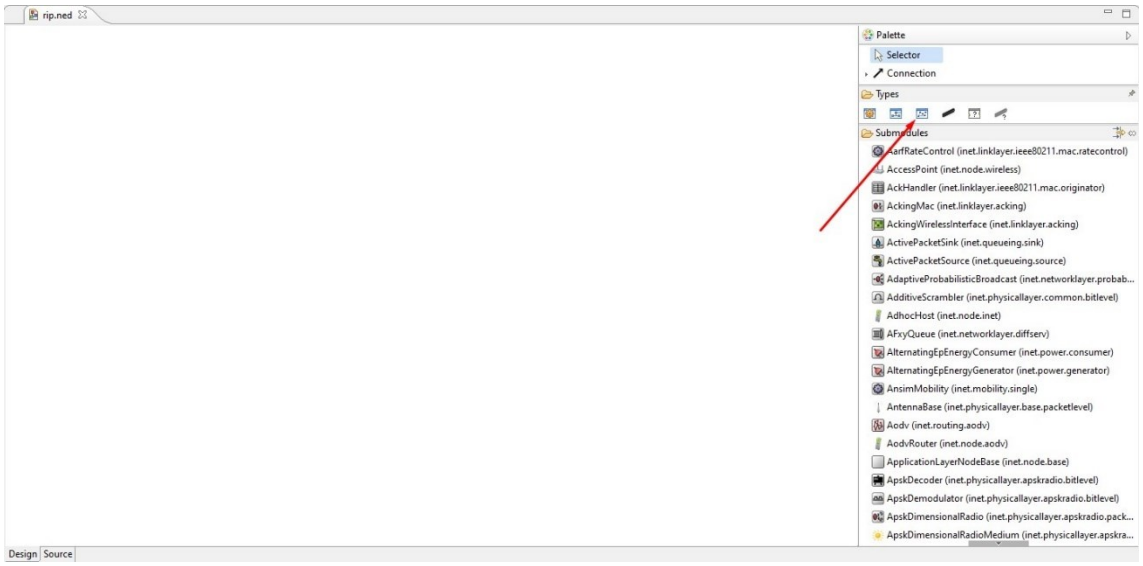
6.1.2 Vytvoření nového projektu

Založte nový projekt v simulačním prostředí OMNeT++: File > New > OMNeT++ project > „Název souboru" > empty project.

Po spuštění je třeba přidat referenci na soubory z INET Framework. V project Exploreru najděte vámi vytvořený projekt, klikněte pravým tlačítkem > properties > Project References > zvolte inet > potvrďte.

6.1.3 Vytvoření .ned modelu pro síť

Ve sloupci Project Explorer najděte váš projekt, klikněte pravým tlačítkem > New > Network Description File (NED) > File name: rip.ned > Empty NED file > Finish



Obrázek 6.2: Vytvoření sítě v NED souboru

Podle obrázku 6.2 vytvořte v nově vytvořeném NED souboru síť, kterou následně přejmenujte kliknutím pravým tlačítkem do šedého pole > rename > „rip".

Nyní může čtenář využít buď grafického „Design" módu a ve sloupci „Submodules" dohledat veškeré potřebné prvky, které jsou zadány, nebo využít módu zdrojového „Source" kódu, kde vloží následující zdrojový kód, který zajistí usazení veškerých požadovaných zařízení, včetně jejich vzájemného propojení, (viz příloha A).

Nyní, přepnete-li zpátky do design módu, uvidíte síť zapojenou identicky se zadáním.

6.1.4 Konfigurace rozhraní

Konfiguraci všech rozhraní sítě zajišťuje modul Ipv4NetworkConfigurator. Tento modul na vstup příjme .xml soubor, ve kterém přiřadíte všem prvkům IP adresy. Prvně tento .xml soubor vytvořte: klikněte pravým tlačítkem na váš projekt ve sloupci Project Explorer, pak New > File > File name: „Config.xml" a potvrďte. V nově vytvořeném souboru pak přiřaďte IP adresy na jednotlivá rozhraní, (viz příloha B).

6.1.5 Inicializace pomocí omnetpp.ini souboru

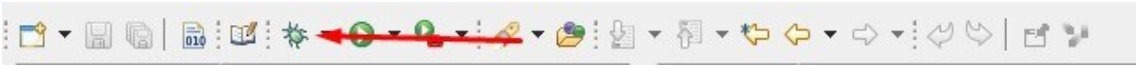
Nastavení jednotlivých parametrů simulace a přiřazení všech konfiguračních souborů konfiguračním modulům zajistí inicializační soubor omnetpp.ini. Zde pomocí numApps vytvoříte jednoduchý ping, kterým otestujete funkčnost sestavené topologie. Ping bude odesílán až od určitého času běhu simulace z důvodu plnění směrovacích tabulek směrovačů. Soubor vytvořte totožným způsobem jako předešlé soubory: : New > Initialization File > File name: omnetpp.ini > Empty Ini File > finish. Soubor následně přepněte do módu zdrojového kódu „Source". Nyní proveďte přiřazení vámi vytvořených konfiguračních souborů jednotlivým konfiguračním prvkům, konfiguraci pingu pomocí numApps, zakázání statického směrování, nastavení prvku visualizer, který se postará o zobrazení IP adres na jednotlivých rozhraních v síti a konečně nastavení prvku scenarioManager, který zajistí v konkrétní čas požadované odpojení linky mezi směrovači RA a RB.

```
[General]
network = rip
**.configurator.config = xmldoc("config.xml")
**.configurator.addStaticRoutes = false
**.configurator.addDefaultRoutes = false
*.A.numApps = 1
*.A.app[0].typename = "PingApp"
*.A.app[0].destAddr = "B"
*.A.app[0].startTime = 20s
**.visualizer.interfaceTableVisualizer.displayInterfaceTables =
true
**.visualizer.interfaceTableVisualizer.nodeFilter =
"not(fullPath(*.N*))"
*.scenarioManager.script = xml("<scenario> \
                                    <disconnect t='30' src-
module='RA' dest-module='RB' /> \
                                </scenario>")
```

Konfigurace inicializačního omnetpp.ini souboru

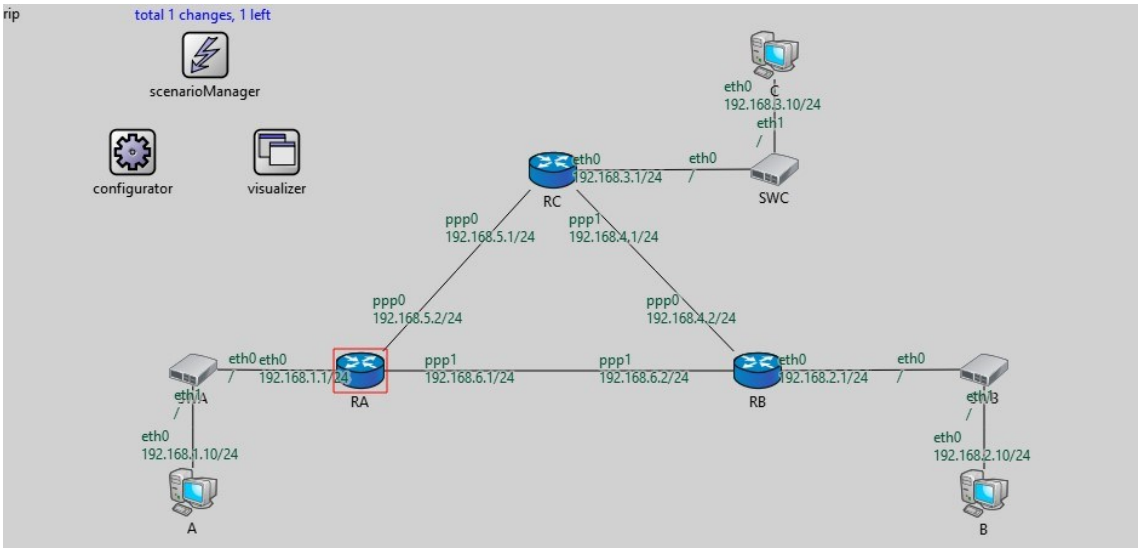
6.1.6 Spuštění simulačního scénáře

Nyní můžete navrženou topologii otestovat. Ke spuštění simulačního okna se uživatel dostane tlačítkem „Debug", které je k nalezení na vrchní liště nástrojů.



Obrázek 6.3: Debug tlačítko

Jestliže byl přesně dodržen postup uvedený výše, měl by se na topologii naskytnout pohled jako je na obrázku 6.4.



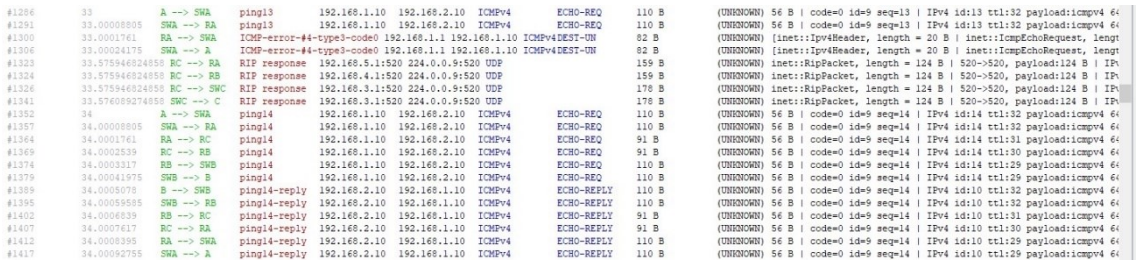
Obrázek 6.4: Topologie v simulačním prostředí

O přehledné zobrazení nastavených adres na jednotlivých rozhraních se postaral IntegratedCanvasVisualizer. Ve spodní části simulačního okna přepněte mód zobrazení na „Show message/packet traffic" (viz obrázek 6.5). Samotnou simulaci pak spusťte „RUN" tlačítkem na horní liště nástrojů. Běh simulace doporučuji urychlit tlačítkem „FAST" a krátce na to simulaci zastavit tlačítkem „STOP".

Obrázek 6.5: Okno zobrazující simulační události

Nyní v okně s událostmi najděte úplný start simulace. Časovou orientaci zajistí sloupec „Time". V úvodu simulace lze pozorovat vzájemná komunikace směrovačů pomocí RIP request a RIP response paketů. Tyto dotazy a následné odpovědi slouží k získání směrovacích informací na jejichž základě pak dochází k plnění směrovacích tabulek směrovačů. Před odesláním prvního pingu dojde k dotazování hosta A na strukturu jeho podsítě pomocí ARP dotazu. Opačný scénář lze pozorovat v podsíti s hostem B, kde se ARP dotazem dotazuje směrovač, který hledá hosta, jemuž náleží daný ping. Strukturu těchto zpráv lze vidět ve sloupci „Info". Zásadní změna v běhu simulace pak lze pozorovat ve třicáté vteřině, kdy dojde k odpojení linky mezi směrovači RA a RB. Posléze dojde ke snaze hosta A o komunikaci s hostem B, která je ovšem bezvýsledná.

Úspěšná komunikace mezi hostem A a hostem B probíhá po obnově směrovacích informací ve směrovacích tabulkách.



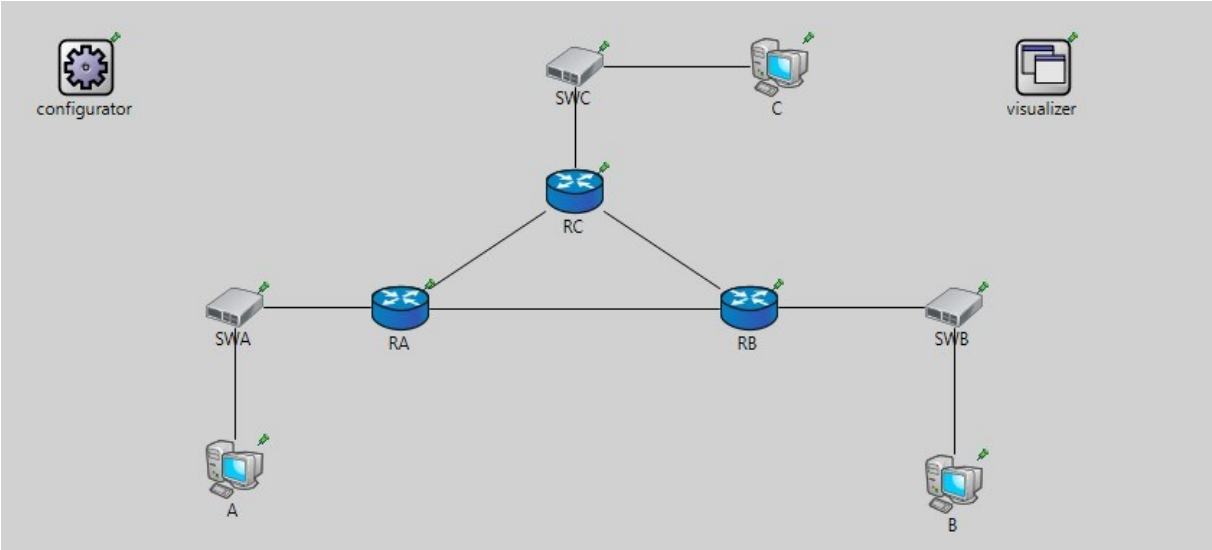
Obrázek 6.6: Komunikace mezi hosty po obnovení směrovacích informací

6.2 Implementace směrovacího protokolu OSPF

Úloha čtenáře provede konfigurací veškerých potřebných souborů při tvorbě simulačního scénáře. S tímto vysvětlením bude částečně poodhalen i princip směrování směrovacího protokolu OSPF. Zadání a jeho následná realizace je právě zaměřeno na metriku tohoto směrovacího protokolu a snaží se poukázat na chování směrovačů, které je při implementaci OSPF ovlivněno cenou (cost) daných spojů.

6.2.1 Zadání

V simulačním prostředí OMNeT++ navrhněte topologii podle obrázku 6.7. Při návrhu využijte předpřipravené prvky OspfRouter, StandardHost, EtherSwitch, IntegratedCanvasVisualizer, Ipv4NetworkConfigurator ve sloupci se submoduly. Dodržte zadaný způsob pojmenování všech prvků podle zadání. Nakonfigurujte zařízení pomocí Ipv4NetworkConfigurator. Pro směrování využijte směrovací protokol OSPF. Správnost navržené topologie otestujte využitím PingApp pingem mezi zařízeními A, B. Upravte cenu linek tak, aby při následném pingu mezi hosty A,B paket využil spojů se směrovačem RC. Výsledná cesta paketu bude A > SWA > RA > RC > RB > SWB > B..



Obrázek 6.7: schéma zapojení topologie s OSPF protokolem

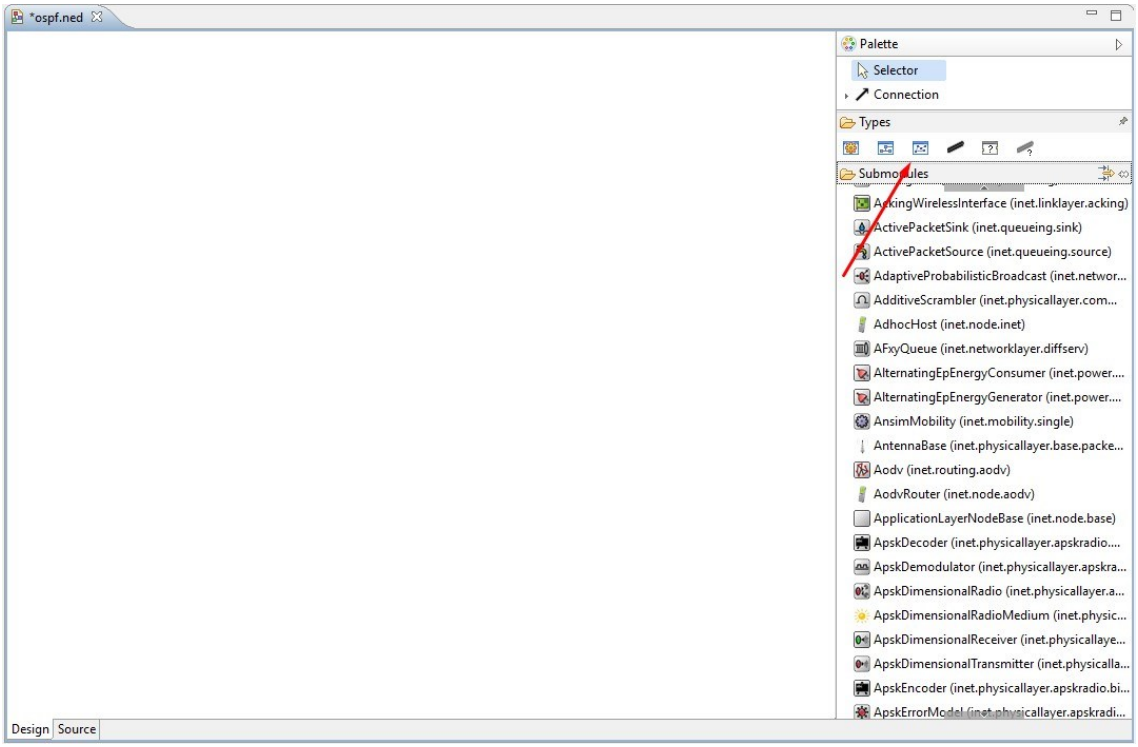
6.2.2 Vytvoření nového projektu

Založte nový projekt v simulačním prostředí OMNeT++: File > New > OMNeT++ project > „Název souboru" > empty project.

Po spuštění je třeba přidat referenci na soubory z INET Framework. V project Exploreru najděte vámi vytvořený projekt, klikněte pravým tlačítkem > properties > Project References > zvolte inet > potvrďte.

6.2.3 Vytvoření .ned modelu pro síť

Ve sloupci Project Explorer najděte váš projekt, klikněte pravým tlačítkem > New > Network Description File (NED) > File name: ospf.ned > Empty NED file > Finish



Obrázek 6.8: Vytvoření sítě v NED souboru

Podle obrázku 6.8 vytvořte v nově vytvořeném NED souboru síť, kterou následně přejmenujte kliknutím pravým tlačítkem do šedého pole > rename > „ospf".

Nyní může čtenář využít buď grafického „Design" módu a ve sloupci „Submodules" dohledat veškeré potřebné prvky, které jsou zadány, nebo využít módu zdrojového „Source" kódu, kde vloží následující zdrojový kód, který zajistí usazení veškerých požadovaných zařízení, včetně jejich vzájemného propojení, (viz příloha C).

Nyní, přepnete-li zpátky do design módu, uvidíte síť zapojenou identicky se zadáním.

6.2.4 Konfigurace rozhraní

Konfiguraci všech rozhraní sítě zajišťuje modul Ipv4NetworkConfigurator. Tento modul na vstup příjme .xml soubor, ve kterém přiřadíte všem prvkům IP adresy. Prvně tento .xml soubor vytvořte: klikněte pravým tlačítkem na váš projekt ve sloupci Project Explorer, pak New > File > File name: „Config.xml" a potvrďte. V nově vytvořeném souboru pak přiřaďte IP adresy na jednotlivá rozhraní, (viz příloha D).

6.2.5 Konfigurace směrovačů v ASconfig.xml souboru

Konfiguraci směrování v síti provedete obdobným způsobem jako při konfiguraci jednotlivých IP adres na síťových rozhraních. Totožným způsobem založte nový soubor, který pojmenujte „ASconfig.xml". V tomto souboru jednoduše zapnete každý „interface" všem směrovačům v síti, (viz příloha E).

6.2.6 Inicializace pomocí omnetpp.ini souboru

Pro nastavení jednotlivých parametrů simulace a přiřazení všech konfiguračních souborů konfiguračnímu modulu slouží inicializační soubor omnetpp.ini. V tomto souboru lze vytvářet různorodé události. Vy zde kromě přiřazení konfiguračních souborů vytvoříte pomocí numApps jednoduchý ping, kterým otestujete funkčnost zapojení topologie. Při vytváření pingu je nutno myslet na fakt, že směrovače potřebují čas na poznávání sítě a na navázání sousedských vztahů při OSPF směrování. Proto je na místě ping posílat až od určitého času. Soubor vytvořte totožným způsobem jako předešlé soubory: New > Initialization File > File name: omnetpp.ini > Empty Ini File > finish. Soubor posléze přepněte do módu zdrojového kódu „Source". Nyní provedete přiřazení vámi vytvořených konfiguračních souborů jednotlivým konfiguračním prvkům, konfiguraci pingu pomocí numApps a konečně nastavení prvku visualizer, který se postará o zobrazení IP adres na jednotlivých rozhraních v síti.

```
[General]
network = Ospf
**.configurator.config = xmldoc("config.xml")
**.configurator.addStaticRoutes = false
**.configurator.addDefaultRoutes = false
**.ospf.ospfConfig = xmldoc("ASconfig.xml")
*.A.numApps = 1
*.A.app[0].typename = "PingApp"
*.A.app[0].destAddr = "B"
*.A.app[0].startTime = 20s
**.visualizer.interfaceTableVisualizer.displayInterfaceTables =
true
```

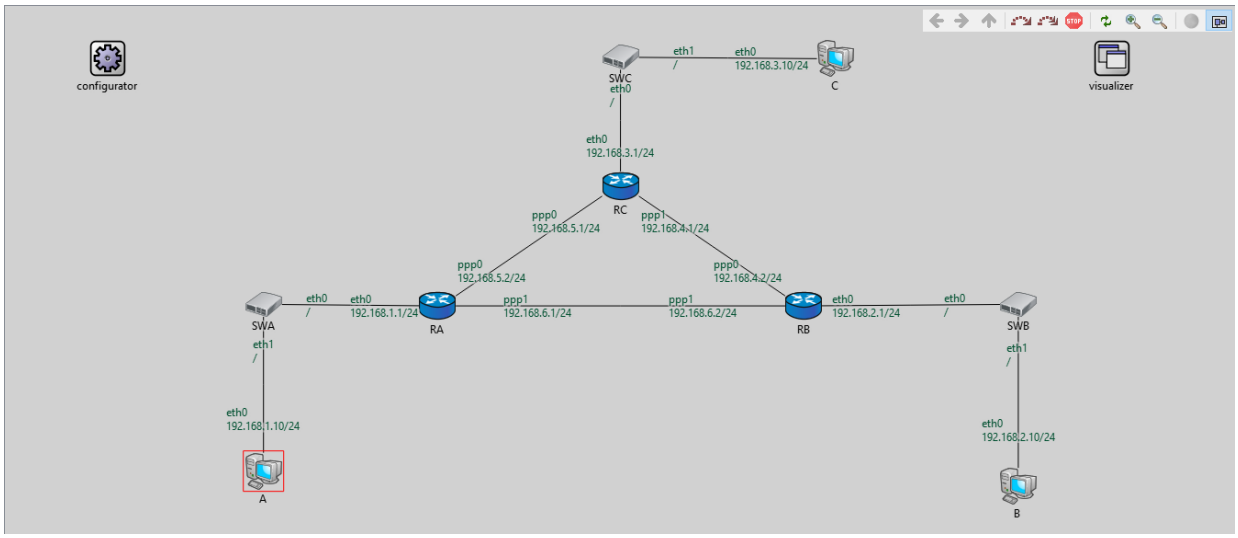
```
**.visualizer.interfaceTableVisualizer.nodeFilter =
"not(fullPath(*.N*))"
```

Konfigurace inicializačního omnetpp.ini souboru

6.2.7 První otestování topologie

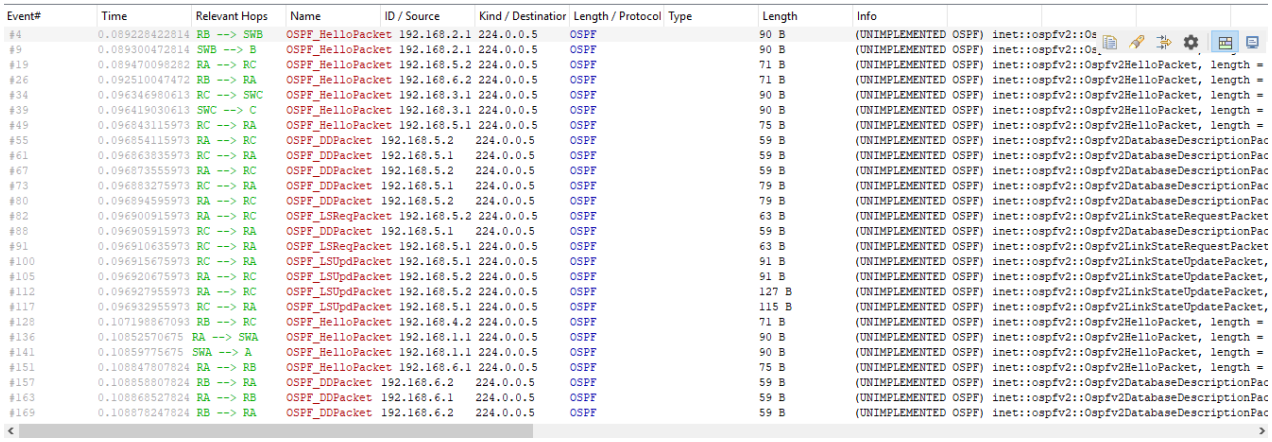
Nyní můžete navrženou topologii otestovat. Ke spuštění simulačního okna se uživatel dostane tlačítkem „Debug", které je k nalezení na vrchní liště nástrojů (viz obrázek 6.3).

Po úspěšném spuštění projektu se dostáváme k oknu samotné simulace. Jestliže byly dodrženy všechny kroky správně, měl by se naskytnout pohled na topologii, jaký je k vidění na obrázku 6.9.



Obrázek 6.9: Topologie v simulačním prostředí

Modul IntegratedCanvasVisualizer se postaral o přehledné zobrazení IP adres na každém aktivním rozhraní. To může být nadmíru užitečné v případě výskytu chyby. Následný simulační scénář pustíme tlačítkem „RUN" na vrchní liště nástrojů. Doporučuji simulaci urychlit tlačítkem „FAST" a krátce potom zastavit tlačítkem „STOP". V běžící simulaci lze sledovat, jakým způsobem si směrovače předávají Hello pakety, díky kterým následně uzavírají sousedství. Veškerá komunikace je vidět ve spodním okně se záznamy událostí (viz obrázek 6.10).



Event#	Time	Relevant Hops	Name	ID / Source	Kind / Destinatior	Length / Protocol	Type	Length	Info
#4	0.089228422814	RB --> SWB	OSPF_HelloPacket	192.168.2.1	224.0.0.5	OSPF		90 B	(UNIMPLEMENTED OSPF) inet::ospfv2::Os
#9	0.089300472814	SWB --> B	OSPF_HelloPacket	192.168.2.1	224.0.0.5	OSPF		90 B	(UNIMPLEMENTED OSPF) inet::ospfv2::Os
#19	0.089470098282	RA --> RC	OSPF_HelloPacket	192.168.5.2	224.0.0.5	OSPF		71 B	(UNIMPLEMENTED OSPF) inet::ospfv2::Ospfv2HelloPacket, length =
#26	0.092510047472	RB --> RA	OSPF_HelloPacket	192.168.6.2	224.0.0.5	OSPF		71 B	(UNIMPLEMENTED OSPF) inet::ospfv2::Ospfv2HelloPacket, length =
#34	0.096346980613	RC --> SWC	OSPF_HelloPacket	192.168.3.1	224.0.0.5	OSPF		90 B	(UNIMPLEMENTED OSPF) inet::ospfv2::Ospfv2HelloPacket, length =
#39	0.096419030613	SWC --> C	OSPF_HelloPacket	192.168.3.1	224.0.0.5	OSPF		90 B	(UNIMPLEMENTED OSPF) inet::ospfv2::Ospfv2HelloPacket, length =
#49	0.096843115973	RC --> RA	OSPF_HelloPacket	192.168.5.1	224.0.0.5	OSPF		75 B	(UNIMPLEMENTED OSPF) inet::ospfv2::Ospfv2HelloPacket, length =
#55	0.096854115973	RA --> RC	OSPF_DDPacket	192.168.5.2	224.0.0.5	OSPF		59 B	(UNIMPLEMENTED OSPF) inet::ospfv2::Ospfv2DatabaseDescriptionPac
#61	0.096863835973	RC --> RA	OSPF_DDPacket	192.168.5.1	224.0.0.5	OSPF		59 B	(UNIMPLEMENTED OSPF) inet::ospfv2::Ospfv2DatabaseDescriptionPac
#67	0.096873555973	RA --> RC	OSPF_DDPacket	192.168.5.2	224.0.0.5	OSPF		59 B	(UNIMPLEMENTED OSPF) inet::ospfv2::Ospfv2DatabaseDescriptionPac
#73	0.096883275973	RC --> RA	OSPF_DDPacket	192.168.5.1	224.0.0.5	OSPF		79 B	(UNIMPLEMENTED OSPF) inet::ospfv2::Ospfv2DatabaseDescriptionPac
#80	0.096894595973	RA --> RC	OSPF_DDPacket	192.168.5.2	224.0.0.5	OSPF		79 B	(UNIMPLEMENTED OSPF) inet::ospfv2::Ospfv2DatabaseDescriptionPac
#82	0.096900915973	RA --> RC	OSPF_LSReqPacket	192.168.5.2	224.0.0.5	OSPF		63 B	(UNIMPLEMENTED OSPF) inet::ospfv2::Ospfv2LinkStateRequestPacket
#88	0.096905915973	RC --> RA	OSPF_DDPacket	192.168.5.1	224.0.0.5	OSPF		59 B	(UNIMPLEMENTED OSPF) inet::ospfv2::Ospfv2DatabaseDescriptionPac
#91	0.096910635973	RC --> RA	OSPF_LSReqPacket	192.168.5.1	224.0.0.5	OSPF		63 B	(UNIMPLEMENTED OSPF) inet::ospfv2::Ospfv2LinkStateRequestPacket
#100	0.096915675973	RC --> RA	OSPF_LSUpdPacket	192.168.5.1	224.0.0.5	OSPF		91 B	(UNIMPLEMENTED OSPF) inet::ospfv2::Ospfv2LinkStateUpdatePacket,
#105	0.096920675973	RA --> RC	OSPF_LSUpdPacket	192.168.5.2	224.0.0.5	OSPF		91 B	(UNIMPLEMENTED OSPF) inet::ospfv2::Ospfv2LinkStateUpdatePacket,
#112	0.096927955973	RA --> RC	OSPF_LSUpdPacket	192.168.5.2	224.0.0.5	OSPF		127 B	(UNIMPLEMENTED OSPF) inet::ospfv2::Ospfv2LinkStateUpdatePacket,
#117	0.096932955973	RC --> RA	OSPF_LSUpdPacket	192.168.5.1	224.0.0.5	OSPF		115 B	(UNIMPLEMENTED OSPF) inet::ospfv2::Ospfv2LinkStateUpdatePacket,
#128	0.107198867093	RB --> RC	OSPF_HelloPacket	192.168.4.2	224.0.0.5	OSPF		71 B	(UNIMPLEMENTED OSPF) inet::ospfv2::Ospfv2HelloPacket, length =
#136	0.108525706750	RA --> SWA	OSPF_HelloPacket	192.168.1.1	224.0.0.5	OSPF		90 B	(UNIMPLEMENTED OSPF) inet::ospfv2::Ospfv2HelloPacket, length =
#141	0.108597756750	SWA --> A	OSPF_HelloPacket	192.168.1.1	224.0.0.5	OSPF		90 B	(UNIMPLEMENTED OSPF) inet::ospfv2::Ospfv2HelloPacket, length =
#151	0.108847807824	RA --> RB	OSPF_HelloPacket	192.168.6.1	224.0.0.5	OSPF		75 B	(UNIMPLEMENTED OSPF) inet::ospfv2::Ospfv2HelloPacket, length =
#157	0.108858807824	RB --> RA	OSPF_DDPacket	192.168.6.2	224.0.0.5	OSPF		59 B	(UNIMPLEMENTED OSPF) inet::ospfv2::Ospfv2DatabaseDescriptionPac
#163	0.108868527824	RA --> RB	OSPF_DDPacket	192.168.6.1	224.0.0.5	OSPF		59 B	(UNIMPLEMENTED OSPF) inet::ospfv2::Ospfv2DatabaseDescriptionPac
#169	0.108878247824	RB --> RA	OSPF_DDPacket	192.168.6.2	224.0.0.5	OSPF		59 B	(UNIMPLEMENTED OSPF) inet::ospfv2::Ospfv2DatabaseDescriptionPac

Obrázek 6.10: Okno se záznamy událostí proběhlé simulace

Kromě směrovacích informací lze v záznamech dohledat rovněž i ARP dotazy a odpovědi odeslané mezi hosty a jejich výchozími směrovači. Ty proběhnou před odesláním prvního pingu. První ping lze dohledat v záznamech podle času, který jste mu nastavili v inicializačním souboru. Souhrnně tedy můžete úplně první komunikaci mezi hostem A, a hostem B, včetně potřebných ARP dotazů a odpovědí pozorovat ve dvacáté vteřině běhu simulace.

6.2.8 Úprava topologie pro potřebu zadání

Po úspěšném otestování komunikace mezi dvěma hosty můžete pokročit k úpravě topologie tak, aby splňovala zadanou podmínku. Zadání požaduje, aby ping mezi hostem A a hostem B využil spojení se směrovačem RC. Výsledná cesta pingu tedy bude A > SWA > RA > RC > RB > SWB > B. Pro tuto úpravu je nutné se zamyslet, jakým způsobem funguje směrování při použití směrovacího protokolu OSPF. Konkrétně jakým způsobem je řešená metrika, tedy kritérium, podle kterého směrovače vybírají nejvýhodnější cestu paketu ze zdroje do cíle. Konkrétně protokol OSPF používá metriku označovanou jako cena (cost). Čím menší cena, tím má cesta lepší metriku a při rozhodování směrovače bude preferována. Cenu lze spočítat podle vzorce: $cost = \frac{100000000}{\text{bandwidth}}$. Užitím tohoto předpokladu budete vycházet při úpravě topologie. Úkolem tedy je upravit cenu linek mezi směrovači tak, aby cesta přes směrovač RC byla cenově výhodnější, než přímá cesta mezi směrovači RA a RB. Vámi vytvořená topologie mezi směrovači používá point to point linek s propustností 100Mbps a zpožděním 5µs. Vytvoříte tedy novou linku, která bude výrazně méně výhodná než dosud používané linky, a nahradíme jí linku mezi směrovači RA a RB. To provedeme nejprve deklarací nového p2p spoje v ospf.ned souboru. Mezi prvky types tedy přidáme následný kus kódu:

```
channel PppLink10M extends DatarateChannel
        {
            delay = 5us;
            datarate = 10Mbps;
        }
```

Deklarace nové linky

Nyní už zbývá pouhé nahrazení spoje ve svazku „connections" mezi směrovači RA a RB vámi nově deklarovanou linkou s propustností 10Mbps. Spojení mezi směrovači tedy bude nově vypadat následovně:

```
RA.pppg++ <--> PppLink10M <--> RB.pppg++;
```

Nahrazená linka mezi směrovači RA a RB

V poslední řadě zbývá topologii otestovat. To provedete stejně jako v kapitole 6.2.7. Pokud jste postupovali přesně podle uvedeného postupu, měli byste v záznamu události vidět následné chování při prvním pingu mezi směrovači RA a RB.

#820	20	A --> SWA	arpREQ	0A-AA-00-00-00-04	FF-FF-FF-FF-FF-FF	ARP	REQUEST
#825	20.00005765	SWA --> RA	arpREQ	0A-AA-00-00-00-04	FF-FF-FF-FF-FF-FF	ARP	REQUEST
#833	20.0001153	RA --> SWA	arpREPLY	0A-AA-00-00-00-01	0A-AA-00-00-00-04	ARP	REPLY
#839	20.00017295	SWA --> A	arpREPLY	0A-AA-00-00-00-01	0A-AA-00-00-00-04	ARP	REPLY
#847	20.0002306	A --> SWA	ping0	192.168.1.10	192.168.2.10	ICMPv4	ECHO-REQ
#853	20.00031865	SWA --> RA	ping0	192.168.1.10	192.168.2.10	ICMPv4	ECHO-REQ
#860	20.0004067	RA --> RC	ping0	192.168.1.10	192.168.2.10	ICMPv4	ECHO-REQ
#865	20.00041898	RC --> RB	ping0	192.168.1.10	192.168.2.10	ICMPv4	ECHO-REQ
#870	20.00043126	RB --> SWB	arpREQ	0A-AA-00-00-00-02	FF-FF-FF-FF-FF-FF	ARP	REQUEST
#875	20.00048891	SWB --> B	arpREQ	0A-AA-00-00-00-02	FF-FF-FF-FF-FF-FF	ARP	REQUEST
#883	20.00054656	B --> SWB	arpREPLY	0A-AA-00-00-00-05	0A-AA-00-00-00-02	ARP	REPLY
#889	20.00060421	SWB --> RB	arpREPLY	0A-AA-00-00-00-05	0A-AA-00-00-00-02	ARP	REPLY
#897	20.00066186	RB --> SWB	ping0	192.168.1.10	192.168.2.10	ICMPv4	ECHO-REQ
#903	20.00074991	SWB --> B	ping0	192.168.1.10	192.168.2.10	ICMPv4	ECHO-REQ
#913	20.00083796	B --> SWB	arpREQ	0A-AA-00-00-00-05	FF-FF-FF-FF-FF-FF	ARP	REQUEST
#919	20.00089561	SWB --> RB	arpREQ	0A-AA-00-00-00-05	FF-FF-FF-FF-FF-FF	ARP	REQUEST
#927	20.00095326	RB --> SWB	arpREPLY	0A-AA-00-00-00-02	0A-AA-00-00-00-05	ARP	REPLY
#933	20.00101091	SWB --> B	arpREPLY	0A-AA-00-00-00-02	0A-AA-00-00-00-05	ARP	REPLY
#941	20.00106856	B --> SWB	ping0-reply	192.168.2.10	192.168.1.10	ICMPv4	ECHO-REPLY
#947	20.00115661	SWB --> RB	ping0-reply	192.168.2.10	192.168.1.10	ICMPv4	ECHO-REPLY
#954	20.00124466	RB --> RC	ping0-reply	192.168.2.10	192.168.1.10	ICMPv4	ECHO-REPLY
#959	20.00125694	RC --> RA	ping0-reply	192.168.2.10	192.168.1.10	ICMPv4	ECHO-REPLY
#964	20.00126922	RA --> SWA	ping0-reply	192.168.2.10	192.168.1.10	ICMPv4	ECHO-REPLY
#969	20.00135727	SWA --> A	ping0-reply	192.168.2.10	192.168.1.10	ICMPv4	ECHO-REPLY

Obrázek 6.11: Ping mezi směrovači RA a RB s užitím směrovače RC

Závěr

Cílem této bakalářské práce je popis simulačního systému OMNeT++, jeho rozšíření INET Framework a vytvoření dvou simulačních úloh využívajících tento framework. Vytvořené simulační úlohy formou laboratorního cvičení budou v příštích letech využity ve cvičení odborného předmětu. Tyto laboratorní úlohy studentům poslouží jako podrobný manuál pro tvorbu simulačních scénářů v simulačním prostředí OMNeT++. Úlohy studentům z části představí funkci užitých směrovacích protokolů.

První navržená úloha věnující se směrování v síti s užitím RIP směrovacího protokolu studenta krok za krokem provede vytvořením simulačního scénáře. Na studenta neklade přílišné nároky, co se znalostí směrování a směrovacích protokolů týče. Snaha v této úloze je zaměřena zejména na představení a následné pochopení myšlenky, kterou užívá simulační prostředí OMNeT++. Závěr této kapitoly studenty dovede k zobrazení simulačního průběhu, na kterém je taktéž v krátkosti vysvětleno chování studentem vytvořené topologie.

Druhá navržená úloha věnující se směrování v síti s užitím OSPF směrovacího protokolu studenta opět postupně provede implementací navrženého simulačního scénáře. Od předchozí úlohy se zadání drobně liší složitostí. Druhá úloha byla navržena tak, aby kladla nárok na znalosti v oblasti chování OSPF směrovacího protokolu. Řešení navrženého zadání tedy vychází ze znalosti metriky OSPF protokolu. Myšlenka je opět studentům detailně vysvětlena v průběžném postupu laboratorní úlohou.

Obě úlohy tedy určitým způsobem představují práci s prostředím, ale rovněž způsob směrování a chování jednotlivých prvků, které jsou denně užívány v datových sítích.

V průběhu zpracování této práce jsem si povšimnul, jakým způsobem se INET Framework vyvíjí. Změny provedené vývojáři v implementaci jednotlivých modulů razantně mění práci s těmito moduly. Proto je dost reálná obava, že v průběhu pár let se tato práce stane práci neaktuální a bez provedení nutných změn v implementaci navržených simulačních scénářů budou tyto navržené laboratorní úlohy nepoužitelné ve cvičeních daného odborného předmětu. Simulační prostředí OMNeT++ v porovnání s velice rozšířeným simulátorem Cisco Packet Tracer pozbývá intuitivnosti a případná práce s prostředím může být pro uživatele náročnější. Na druhou stranu uživateli poskytuje širší možnosti při práci na jednotlivých simulačních scénářích než jiná simulační prostředí.

Použitá literatura

[1] Modeling and Tools for Network Simulation. WEHRLE, Klaus, Mesut GÜNE¸S a James GROSS. Modeling and Tools for Network Simulation. Springer-Verlag Berlin Heidelberg, 2010. ISBN 978-3-642-12331-3.

[2] OMNeT++ Simulation Manual. Omnetpp [online]. [cit. 2020-04-20]. Dostupné z: https://doc.omnetpp.org/omnetpp/manual/#cha:simple-modules

[3] ZDAŘIL, Michal. Simulační prostředí OMNeT++ [online]. Ostrava, 2009 [cit. 2020-04-20]. Dostupné z: http://hdl.handle.net/10084/75400. Bakalářská práce. Vysoká škola báňská - Technická univerzita Ostrava.

[4] OMNeT++ Simulation Manual. Omnetpp [online]. [cit. 2020-04-20]. Dostupné z: https://doc.omnetpp.org/omnetpp/manual/#sec:introduction:what-is-omnetpp

[5] OMNeT++ Simulation Manual. Omnetpp [online]. [cit. 2020-04-20]. Dostupné z: https://doc.omnetpp.org/omnetpp/manual/#sec:simple-modules:events-in-opp

[6] OMNeT++ Simulation Manual. Omnetpp [online]. [cit. 2020-04-20]. Dostupné z: https://doc.omnetpp.org/omnetpp/manual/#cha:ned-lang

[7] What Is INET Framework. INET Framework [online]. [cit. 2020-04-20]. Dostupné z: https://inet.omnetpp.org/Introduction

[8] Getting Familiar with INET. INET Framework [online]. [cit. 2020-04-20]. Dostupné z: https://inet.omnetpp.org/docs/users-guide/ch-usage.html#getting-familiar-with-inet

[9] Built-in Network Nodes and Other Top-Level Modules. INET Framework [online]. [cit. 2020-04-20]. Dostupné z: https://inet.omnetpp.org/docs/users-guide/ch-networks.html#built-in-network-nodes-and-other-top-level-modules

[10] Router. INET Framework [online]. [cit. 2020-04-20]. Dostupné z: https://doc.omnetpp.org/inet/api-current/neddoc/inet.node.inet.Router.html

[11] Network Interfaces. INET Framework [online]. [cit. 2020-04-20]. Dostupné z: https://inet.omnetpp.org/docs/users-guide/ch-network-interfaces.html#network-interfaces

[12] Built-in Network Interfaces. INET Framework [online]. [cit. 2020-04-20]. Dostupné z: https://inet.omnetpp.org/docs/users-guide/ch-network-interfaces.html#built-in-network-interfaces

[13] Ethernet Interface. INET Framework [online]. [cit. 2020-04-20]. Dostupné z: https://inet.omnetpp.org/docs/users-guide/ch-ethernet.html#ethernet-interface

[14] The 802.11 Model. INET Framework [online]. [cit. 2020-04-20]. Dostupné z: https://inet.omnetpp.org/docs/users-guide/ch-80211.html#the-802-11-model

[15] Ieee80211Interface. INET Framework [online]. [cit. 2020-04-20]. Dostupné z: https://doc.omnetpp.org/inet/api-current/neddoc/inet.linklayer.ieee80211.Ieee80211Interface.html

[16] Internet Routing. INET Framework [online]. [cit. 2020-04-20]. Dostupné z: https://inet.omnetpp.org/docs/users-guide/ch-routing.html#internet-routing

[17] RIP. INET Framework [online]. [cit. 2020-04-20]. Dostupné z: https://inet.omnetpp.org/docs/users-guide/ch-routing.html#rip

[18] OSPF. INET Framework [online]. [cit. 2020-04-20]. Dostupné z: https://inet.omnetpp.org/docs/users-guide/ch-routing.html#ospf

[19] BGP. INET Framework [online]. [cit. 2020-04-20]. Dostupné z: https://inet.omnetpp.org/docs/users-guide/ch-routing.html#bgp

[20] OMNeT++ Installation Guide. Omnetpp [online]. [cit. 2020-04-20]. Dostupné z: https://doc.omnetpp.org/omnetpp/InstallGuide.pdf

[21] OMNeT++ User Guide. Omnetpp [online]. [cit. 2020-04-20]. Dostupné z: https://doc.omnetpp.org/omnetpp/UserGuide.pdf

Seznam příloh

Příloha A: *Kód konfigurace topologie v rip.ned souboru*

```
import inet.node.ethernet.EtherSwitch;
import inet.node.inet.StandardHost;
import inet.node.rip.RipRouter;
import ned.DatarateChannel;
import inet.node.ethernet.Eth10M;
import
inet.networklayer.configurator.ipv4.Ipv4NetworkConfigurator;
import inet.visualizer.integrated.IntegratedCanvasVisualizer;
import inet.common.scenario.ScenarioManager;
network rip
{
    @display("bgb=944,474");
    types:
        channel PppLink10M extends DatarateChannel
        {
            delay = 5us;
            datarate = 10Mbps;
        }
    submodules:
        RA: RipRouter {
            @display("p=297,304");
            gates:
                ethg[1];
        }
        RB: RipRouter {
            @display("p=625,304");
            gates:
                ethg[1];
        }
        RC: RipRouter {
            @display("p=456,139");
```

```
        gates:
            ethg[1];
    }
    SWA: EtherSwitch {
        @display("p=159,303");
    }
    SWC: EtherSwitch {
        @display("p=639,138");
    }
    SWB: EtherSwitch {
        @display("p=812,303;b=10,12");
    }
    A: StandardHost {
        @display("p=158.79001,404.085");
        gates:
            ethg[1];
    }
    B: StandardHost {
        @display("p=812,405");
        gates:
            ethg[1];
    }
    C: StandardHost {
        @display("p=639,44");
        gates:
            ethg[1];
    }
    configurator: Ipv4NetworkConfigurator {
        @display("p=109,124");
    }
    visualizer: IntegratedCanvasVisualizer {
```

```
            @display("p=228,124");
        }
        scenarioManager: ScenarioManager {
            @display("p=169.455,43.845");
        }
    connections:
        RA.pppg++ <--> PppLink10M <--> RC.pppg++;
        RB.pppg++ <--> PppLink10M <--> RC.pppg++;
        RA.pppg++ <--> PppLink10M <--> RB.pppg++;

        RA.ethg[0] <--> Eth10M <--> SWA.ethg++;
        RB.ethg[0] <--> Eth10M <--> SWB.ethg++;
        RC.ethg[0] <--> Eth10M <--> SWC.ethg++;

        SWA.ethg++ <--> Eth10M <--> A.ethg[0];
        SWB.ethg++ <--> Eth10M <--> B.ethg[0];
        SWC.ethg++ <--> Eth10M <--> C.ethg[0];

}
```

Příloha B: *Kód konfigurace rozhraní v síti v Config.xml souboru*

```
<?xml version="1.0"?>
     <scenario>
          <interface metric="1" netmask="255.255.255.0"
address="192.168.1.1" names="eth0" hosts="RA"/>
          <interface metric="1" netmask="255.255.255.0"
address="192.168.6.1" names="ppp1" hosts="RA"/>
          <interface metric="1" netmask="255.255.255.0"
address="192.168.5.2" names="ppp0" hosts="RA"/>
          <interface metric="1" netmask="255.255.255.0"
address="192.168.2.1" names="eth0" hosts="RB"/>
          <interface metric="1" netmask="255.255.255.0"
address="192.168.6.2" names="ppp1" hosts="RB"/>
          <interface metric="1" netmask="255.255.255.0"
address="192.168.4.2" names="ppp0" hosts="RB"/>
          <interface metric="1" netmask="255.255.255.0"
address="192.168.5.1" names="ppp0" hosts="RC"/>
          <interface metric="1" netmask="255.255.255.0"
address="192.168.4.1" names="ppp1" hosts="RC"/>
          <interface metric="1" netmask="255.255.255.0"
address="192.168.3.1" names="eth0" hosts="RC"/>
          <interface metric="1" netmask="255.255.255.0"
address="192.168.1.10" names="eth0" hosts="A" mtu="1500"/>
          <interface metric="1" netmask="255.255.255.0"
address="192.168.2.10" names="eth0" hosts="B" mtu="1500"/>
          <interface metric="1" netmask="255.255.255.0"
address="192.168.3.10" names="eth0" hosts="C" mtu="1500"/>
          <route netmask="*" hosts="A" interface="eth0"
destination="*"/>
          <route netmask="*" hosts="B" interface="eth0"
destination="*"/>
          <route netmask="*" hosts="C" interface="eth0"
destination="*"/>
     </scenario>
```

Příloha C: *Kód konfigurace topologie v ospf.ned souboru*

```
import
inet.networklayer.configurator.ipv4.Ipv4NetworkConfigurator;
import inet.visualizer.integrated.IntegratedCanvasVisualizer;
import inet.node.ethernet.EtherSwitch;
import inet.node.inet.StandardHost;
import inet.node.ospfv2.OspfRouter;
import inet.node.ethernet.Eth10M;
import ned.DatarateChannel;

network Ospf
{
    @display("bgb=1248.4875,646.425");
    types:
        channel PppLink100M extends DatarateChannel
        {
            delay = 5us;
            datarate = 100Mbps;
        }
    submodules:
        RA: OspfRouter {
            @display("p=456.30002,330.8175");
            gates:
                ethg[1];
        }
        RB: OspfRouter {
            @display("p=763.03503,330.8175");
            gates:
                ethg[1];
        }
        RC: OspfRouter {
            @display("p=609.66754,230.68501");
```

```
        gates:
            ethg[1];
    }
    A: StandardHost {
        @display("p=310.5375,468.975");
        gates:
            ethg[1];
    }
    B: StandardHost {
        @display("p=943.02,481.65002");
        gates:
            ethg[1];
    }
    C: StandardHost {
        @display("p=789.6525,124.215004");
        gates:
            ethg[1];
    }
    SWA: EtherSwitch {
        @display("p=310.5375,329.55002");
    }
    SWB: EtherSwitch {
        @display("p=943.02,329.55002");
    }
    SWC: EtherSwitch {
        @display("p=609.66754,122.9475");
    }
    configurator: Ipv4NetworkConfigurator {
        @display("p=179.985,124.215004");
    }
    visualizer: IntegratedCanvasVisualizer {
```

```
            @display("p=1021.60504,124.215004");
        }
    connections:
        RA.pppg++ <--> PppLink100M <--> RC.pppg++;
        RB.pppg++ <--> PppLink100M <--> RC.pppg++;
        RA.pppg++ <--> PppLink100M <--> RB.pppg++;

        RA.ethg[0] <--> Eth10M <--> SWA.ethg++;
        RB.ethg[0] <--> Eth10M <--> SWB.ethg++;
        RC.ethg[0] <--> Eth10M <--> SWC.ethg++;

        SWA.ethg++ <--> Eth10M <--> A.ethg[0];
        SWB.ethg++ <--> Eth10M <--> B.ethg[0];
        SWC.ethg++ <--> Eth10M <--> C.ethg[0];
}
```

Příloha D: *Kód konfigurace rozhraní v síti v Config.xml souboru*

```
<?xml version="1.0"?>
     <scenario>
          <interface metric="1" netmask="255.255.255.0"
address="192.168.1.1" names="eth0" hosts="RA"/>
          <interface metric="1" netmask="255.255.255.0"
address="192.168.6.1" names="ppp1" hosts="RA"/>
          <interface metric="1" netmask="255.255.255.0"
address="192.168.5.2" names="ppp0" hosts="RA"/>
          <interface metric="1" netmask="255.255.255.0"
address="192.168.2.1" names="eth0" hosts="RB"/>
          <interface metric="1" netmask="255.255.255.0"
address="192.168.6.2" names="ppp1" hosts="RB"/>
          <interface metric="1" netmask="255.255.255.0"
address="192.168.4.2" names="ppp0" hosts="RB"/>
          <interface metric="1" netmask="255.255.255.0"
address="192.168.5.1" names="ppp0" hosts="RC"/>
          <interface metric="1" netmask="255.255.255.0"
address="192.168.4.1" names="ppp1" hosts="RC"/>
          <interface metric="1" netmask="255.255.255.0"
address="192.168.3.1" names="eth0" hosts="RC"/>
          <interface metric="1" netmask="255.255.255.0"
address="192.168.1.10" names="eth0" hosts="A" mtu="1500"/>
          <interface metric="1" netmask="255.255.255.0"
address="192.168.2.10" names="eth0" hosts="B" mtu="1500"/>
          <interface metric="1" netmask="255.255.255.0"
address="192.168.3.10" names="eth0" hosts="C" mtu="1500"/>
          <route netmask="*" hosts="A" interface="eth0"
destination="*"/>
          <route netmask="*" hosts="B" interface="eth0"
destination="*"/>
          <route netmask="*" hosts="C" interface="eth0"
destination="*"/>
     </scenario>
```

Příloha E: *Kód konfigurace rozhraní směrovačů v ASconfig.xml souboru*

```
<?xml version="1.0"?>
<OSPFASConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:schemaLocation="OSPF.xsd">

  <Router name="RA" RFC1583Compatible="true">
    <BroadcastInterface ifName="eth0" />
    <PointToPointInterface ifName="ppp0"  />
    <PointToPointInterface ifName="ppp1"  />
  </Router>

  <Router name="RB" RFC1583Compatible="true">
    <BroadcastInterface ifName="eth0" />
    <PointToPointInterface ifName="ppp0"  />
    <PointToPointInterface ifName="ppp1"  />
  </Router>

  <Router name="RC" RFC1583Compatible="true">
    <BroadcastInterface ifName="eth0" />
    <PointToPointInterface ifName="ppp0"  />
    <PointToPointInterface ifName="ppp1"  />
  </Router>
</OSPFASConfig>
```